

# Automated Consultation for the Diagnosis of Interplanetary Telecommunications

A.G. Quan, U.M. Schwuttke, J.S. Herstein, J.S. Spagnuolo, and S. Burleigh

Jet Propulsion Laboratory  
California Institute of Technology  
4800 Oak Grove Drive MS 301-270  
Pasadena, CA 91109  
alan@puente.jpl.nasa.gov

## Abstract

SHARP is a knowledge-based system for the diagnosis of problems in NASA's Deep Space Network (DSN) telecommunications system. The telecommunications system provides the means of communication between a spacecraft and operations personnel at Jet Propulsion Laboratory. SHARP analyzes problems that occur in both the on-board spacecraft telecom subsystem and in the DSN. It employs a rule-based expert system to detect and diagnose anomalies and to recommend corrective action. This diagnostic logic is represented in the form of a binary decision tree which is encoded in Prolog. The knowledge base is designed to be easy to maintain and expand, with a one-to-one correspondence between decision tree nodes and rules. The knowledge base is embedded within a C program which contains a telemetry interface and a graphical user-interface. Data is obtained both from on-line sources, such as telemetry and Sequence of Events files, and from the user when the required data is not available from any on-line source. The majority of rules are generic in nature, allowing SHARP to be applied to multiple missions.

## Introduction

SHARP (Spacecraft Health Automated Reasoning Program) is a knowledge-based consultant for diagnosing problems in NASA's Deep Space Network (DSN) telecommunications link in near real-time. This paper describes most recent version of SHARP, which is an operational system used by spacecraft mission controllers; earlier SHARP prototypes are described elsewhere [Martin 1990]. The transition of this system to an operational environment involved the design and implementation of a generic architecture and some application specific customization for its first intended long-term end-user, NASA's Mars Observer mission. Subsequent loss of the Mars Observer spacecraft pro-

vided an opportunity to validate claims for the generality of this tool, with an immediate port to the Galileo mission, (launched in 1989 and scheduled to enter Jupiter orbit in 1995). This port was achieved with less than ten percent of the effort required for the original Mars Observer implementation, as only mission specific components of the system were affected. Unlike previous monitoring and diagnosis applications of this type that have been developed at JPL [Schwuttke 1991, Schwuttke 1994], SHARP was designed to be largely generic, with the major part of its knowledge base and other modules applicable to multiple spacecraft missions.

SHARP has been developed for use by the *multimission* control team at the Jet Propulsion Laboratory (JPL), for detection and analysis of spacecraft and ground system problems. The telecommunication (Telecom) downlink provides a means of communication between the spacecraft and ground, and includes the Telecom subsystem on the spacecraft and components of the DSN. Telemetry data arrives at a DSN station such as the one shown in Figure 1. Stations are located in the U.S, Australia, and Spain. Data from these stations is sent to JPL for monitoring and analysis. There are two kinds of spacecraft telemetry data: the science data, which contains actual data obtained from science instruments on the spacecraft, and the engineering data which provides the health and status of the spacecraft subsystems. There is also a third kind of telemetry known as monitor data, which provides information about the DSN. SHARP concerns itself only with the latter two types of data.

SHARP assists the mission controller responsible for quick-look Telecom analysis in routine operations by automatically detecting degradation in the Telecom link, notifying the mission controller, isolating the cause through interactive consultation, and recommending corrective action. In past interplanetary missions, these capabilities have been provided by a full-time human analyst with expertise in spacecraft telecommunications. More recent missions have not

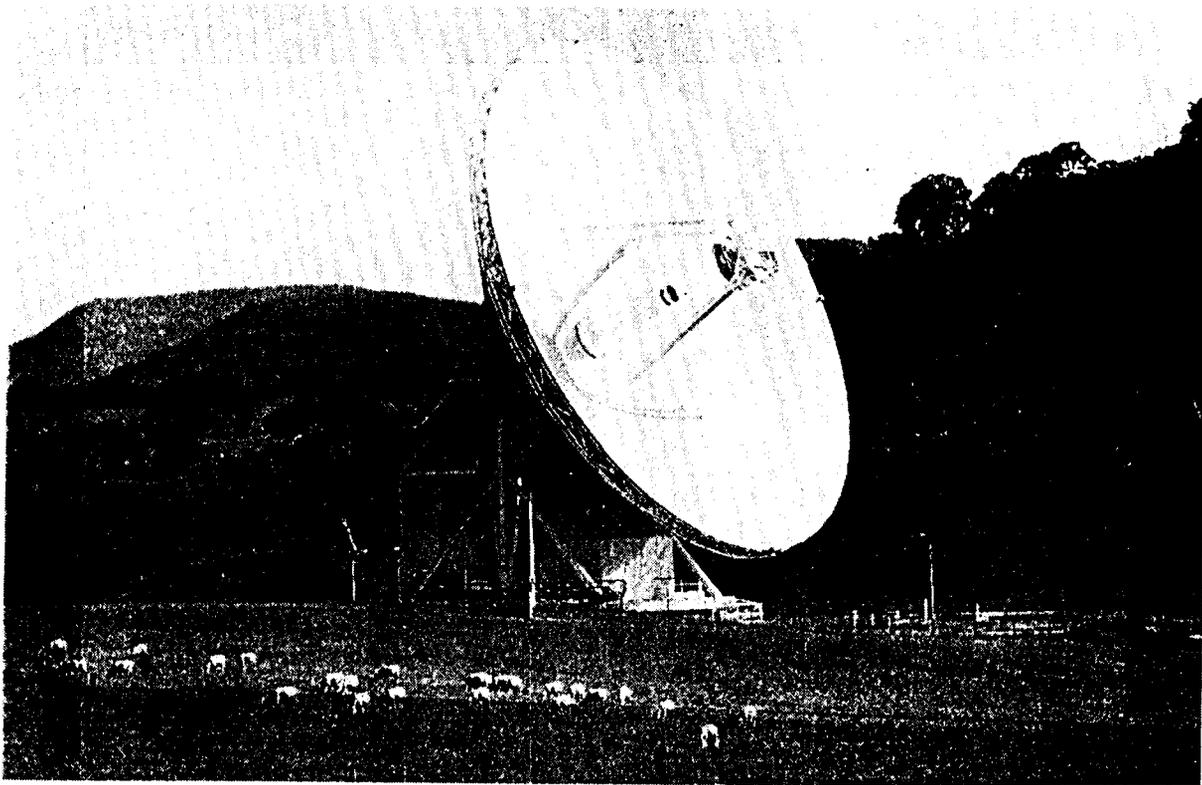


Figure 1. A 34-meter high-efficiency antenna at the DSN Station in Canberra, Australia,

been able to fund full-time telecommunications support. As a result, there has been an increased need for automated and intelligent tools that can substitute for a human specialist by providing interactive consultation to augment the general knowledge of the mission controllers. SHARP answers this need by implementing domain expertise and embedding artificial intelligence techniques for diagnosing anomalies in a near-real-time monitoring system.

### **Problem Description: Mission Control and Telecom Link Analysis**

The responsibility of the Mission Control Team is to coordinate and control spacecraft flight operations, transmitting commands to the spacecraft and performing real-time monitoring of spacecraft performance, the telecommunications link, and the Ground Data System. The Mission Control Team is also responsible for correcting spacecraft problems by issuing commands to the spacecraft. Telecommunication anomalies occur frequently, interrupting the transmission of

data from the spacecraft. Detection and correction of the anomalies requires coordinated monitoring and diagnosis of both the spacecraft and Deep Space Network telecommunications systems. Without SHARP, this activity would still be performed manually, requiring a mission control team member to first detect that a problem exists by monitoring key telemetry values and noticing when they stray outside predetermined tolerances. Once the telemetry symptoms of the problem are discovered, a Telecom domain expert is consulted. The expert attempts to diagnose the cause of the anomaly based upon past experience, detailed knowledge of the telecommunications system, and examination of other related telemetry data as well as non-telemetric data such as the schedule of spacecraft events. After the cause of the anomaly is deduced, the expert must then decide what actions to take in order to correct it, if possible. This process can be rather time consuming, depending on the nature of the anomaly, the availability of the Telecom expert, and whether the initial corrective action chosen by the expert is effective.

Dependence upon a domain expert for anomaly diagnosis and recovery has two major drawbacks. First, this requires that an expert be on call 24 hours a

day, since anomalies can occur at any time. The timeliness of an expert's response to a problem can be critical for recovering important data, or in the extreme case, saving a spacecraft. Secondly, when the experts retire, their skills are lost. The accumulated expertise of mission operations personnel is an important resource which should be preserved rather than recreated every time a senior engineer leaves the flight project. SHARP is designed to serve as an automated Telecom consultant for mission controllers, offering the problem-solving ability of a human expert for well-understood telecommunications anomalies. In this capacity, SHARP allows fast response time in anomaly diagnosis and correction, and preservation of mission expertise for the entire duration of a mission.

## Application Description

SHARP implements Telecom domain expertise and applies artificial intelligence (AI) techniques for diagnosing anomalies in the Telecom downlink. This is accomplished by examining relevant telemetry data from both the DSN and the spacecraft. If an anomaly is detected, SHARP uses knowledge-based analysis to diagnose the problem and provide a recommendation for corrective action. The SHARP system is composed of three primary modules, as shown in Figure 2: a telemetry communication interface to read telemetry from the ground data subsystem, a rule-based expert system to diagnose anomalies, and a graphical user interface (GUI) to display telemetry alarms and diagnostic messages.

### The Telemetry Communication Interface

The telemetry communication interface reads telemetry from the ground data system (GDS) and passes it to the GUI and knowledge base. It is composed of a SHARP-to-GDS telemetry server, an expert system message interface layer, and a GUI message interface layer. The telemetry server obtains raw telemetry data in ASCII from the GDS via a UNIX pipe and converts it to a format for use by the knowledge base and GUI. The data retrieved from the GDS is in the form of text lines which are written to either a first-in-first-out queue (FIFO) or a disk file. The formatted lines can then be distributed to a maximum of five clients. The telemetry interface accepts UNIX socket connections from the expert system and the user interface allowing interprocess communication. When the knowledge base requests data, the expert system message interface layer reads the formatted telemetry data from the serv-

er and passes it to the expert system. The formatted telemetry can also be sent directly to the GUI through a connection to the SHARP-to-GDS telemetry server.

### The Graphical User Interface

The GUI provides the interface between SHARP and the end-user, and has been designed according to the needs of the mission control end-user who is required to perform a large number of functions in addition to monitoring the Telecom link between ground and spacecraft. The user interface is a mouse driven, Motif-based GUI consisting of the three resizable windows shown in Figure 3. These include a status window containing alarm and message status counters and a graphical data quality indicator, an alarm and message summary window, and a utility window for viewing diagnostic messages, telemetry values, and other information. The status window acts as the main window and is the only window visible at start-up. To conserve screen real estate, the window begins in a reduced form displaying Summary and Utility push buttons to pop up corresponding additional windows, a Show All button to display all windows, and a Status button to toggle back and forth between the main window's compressed and expanded states.

When an anomaly is detected by the expert system, the Status button turns red. The color change is accompanied by a beep to alert the user. When the status button is pressed, the window expands to display the Alarm and System Status counters and the data quality indicator. Each counter consists of two fields, one displaying the time of the most recent reset, and the other the number of anomalies the expert system has detected since the last time the counter was reset. When the expert system detects an anomaly or sends a status message to the GUI, the counter fields will turn either red in the Alarm Status display to signal an anomaly or yellow in the System Status display to signal a system message and the corresponding count field will be incremented by one. The user can manually reset a single status counter by selecting the Reset button, thereby resetting the counter to zero, resetting the counter time to the current system time, and converting the counter fields to their normal color. The Ack (acknowledge) buttons change the counter fields back to their normal color only. The Reset and Ack buttons in the Alarm Status display will also change the Status button at the top of the display to its non-alarm color. The Status window also contains a graphical data quality indicator which consists of a four-part scale and slider button whose position and color indicate the quality of the telemetry currently being received (from "Noisy" to "Error Free").

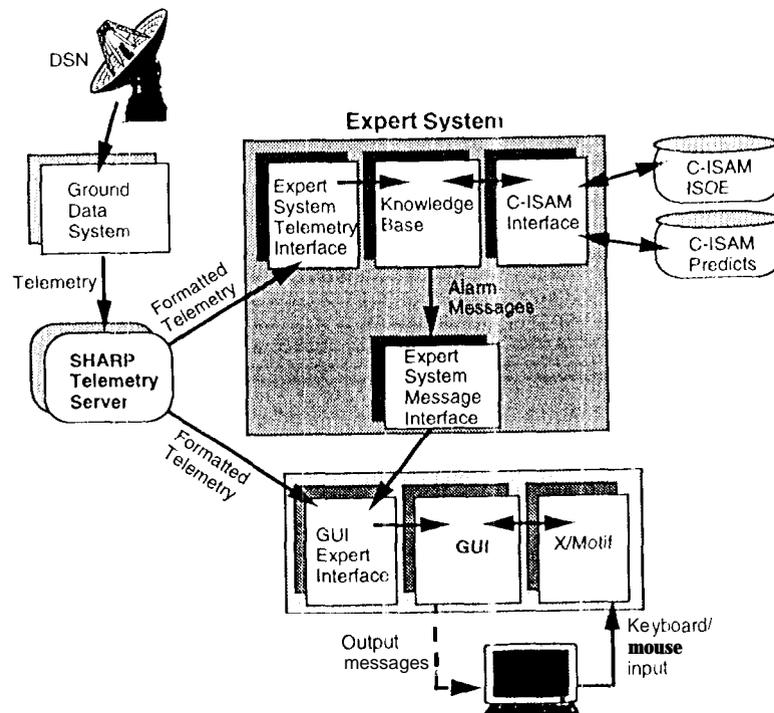


Figure 2. The high-level architecture of the S} IARF' system.

The Summary button pops up the Summary window, which is comprised of two scrollable paned windows that display brief one-line descriptions of the anomaly and informational messages. These messages are provided by the expert system. When the user double clicks on an anomaly message in the alarm message summary window, the Utility window pops up and displays a complete diagnostic message describing the anomaly and a recommendation for corrective action. The user may also view multiple diagnostic messages by selecting one or more anomaly messages and pressing the Diagnosis button, which pops up the Utility window and displays the information in the workspace. The user can also press either the Alarms or Messages buttons in the Summary window to pop up the Utility window and display a complete listing of all the current one-line anomaly or informational messages.

The Utility window is a scrollable user workspace area dedicated to displaying information of current interest to the user, such as diagnostic messages, alarm or system summary lists, or a table of telemetry data. The SOE button at the bottom of the window launches the Sequence of Events (SOE) file viewer and editor, which enable the analyst to see the scheduled spacecraft activities integrated with information from the DSN. The Data button displays a

table of incoming telemetry channel data including columns for the channel type and number, the symbolic description of the channel (mnemonic), the time tag of the telemetry, the engineering value, the data number, and the predicted value (predict) for those channels for which predicts are available.

#### Database Files

Certain data that is required by the knowledge base originates in the predict files and SOE files. Two C-ISAM (indexed sequential access method) database files are maintained, one for predict data and one for SOE data. A set of C functions invoked from the expert system accesses these databases to facilitate the retrieval of information. When SHARP is initialized, the SOE files in the directory of SOE data are processed to extract information concerning specific events that is used by the knowledge base. The data is compounded into a single C-ISAM SOE database file for easy retrieval when the data is requested. Also at initialization, the data in the predict files for each DSN station are transferred into a C-ISAM predict database file. Every ten seconds, the directories containing the SOE data and predict data are checked for added files that are newer than the corresponding C-ISAM file. If a newer SOE file is detected, the current C-ISAM SOE

file is destroyed and re-created using all the SOE files in the directory. If a newer predict file is detected, the relevant contents of that file are added to the C-ISAM predict file.

### The Knowledge Base

The SHARP knowledge base embodies the diagnostic logic of a Telecom expert. This domain knowledge was supplied by a senior Telecom analyst in the form of a binary decision tree. Each time a complete set of telemetry is received from the Ground Data System, the top level node in the decision tree is evaluated. Based upon the value of the data examined at that node, one of two branches is taken to the next node in the tree. This process continues until a leaf node is encountered, at which time the anomaly causing the problem is obtained, along with a recommendation on how to fix the problem.

At each node one or more data values is examined. This data can come from one of several sources, including the telemetry, the predict database, and the SOE database. Sometimes the data is not directly available and must be derived or calculated from other data. Sometimes the data value is not available from any on-line data source (e. g., the weather condition at the DSN station). In this case the user must be asked to supply the value of the data. A pop-up window appears with a question for the user at the top of the window and a menu of possible answers below the question. The user supplies the answer by selecting the menu choice with the mouse and clicking on the "OK" button, or by simply double-clicking on the menu choice.

Whenever an expert system must enter a consultation mode to ask for data from the user, it is important to constrain the questions so that they per-

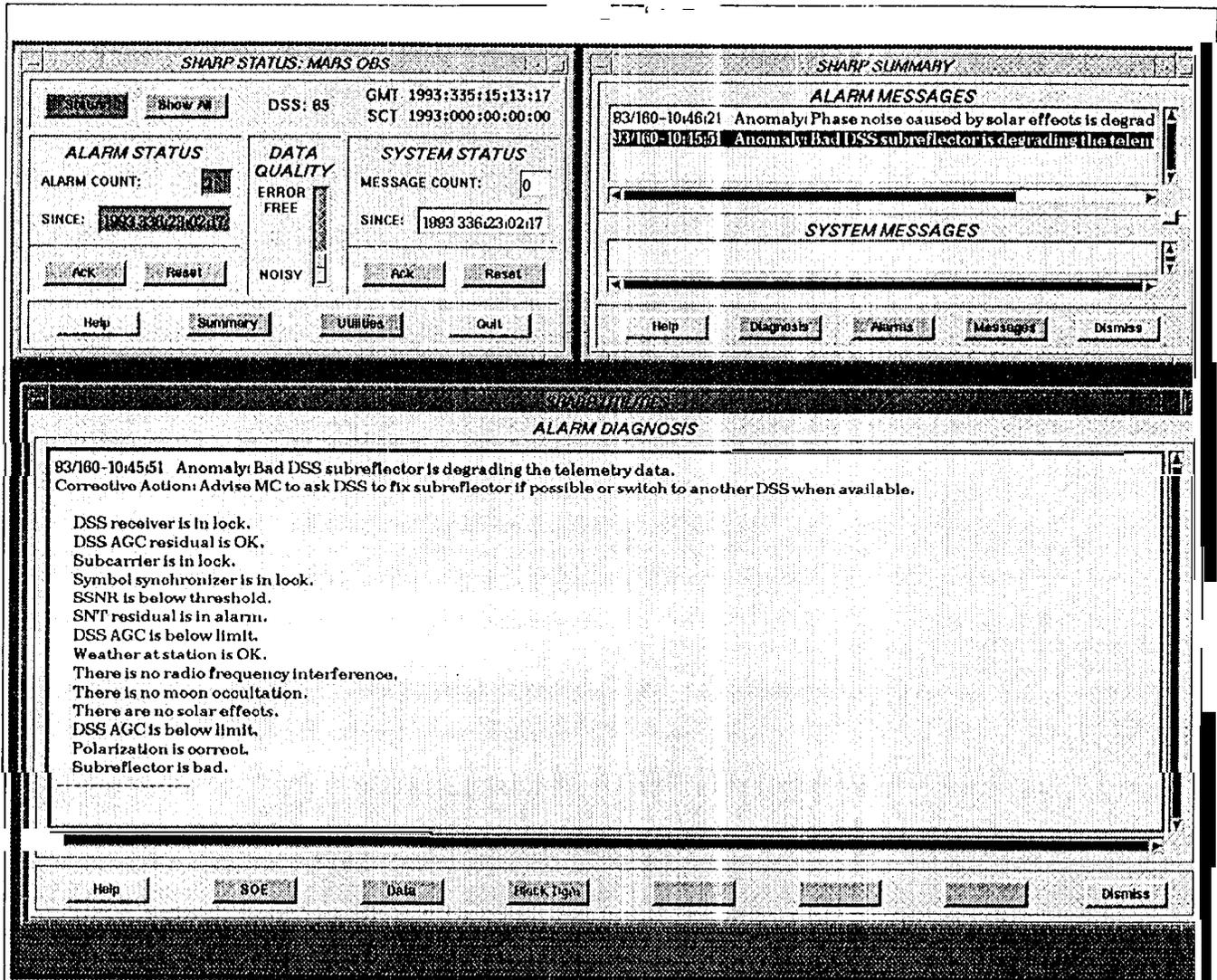


Figure 3. The windows of the graphical user interface.

tain only to the current analysis, rather than asking about data from a decision tree branch that can be excluded from the search because it is irrelevant [Shortliffe 1976]. The same consideration applies to complex calculations that must be performed to transform data into a form that can be properly analyzed (e.g., Fourier analysis). In order to meet real-time constraints, the system must not perform time-consuming calculations of values that are not essential to the current analysis.

In a traditional forward-chained diagnostic system all or most data values must be asserted in working memory prior to the activation of any rules. Subsequently, rule firing is initiated causing relevant rules to be activated by a subset of the data in working memory [Forgy 1982]. Assertion of all data values in SHARP would require asking the user all possible questions and performing all possible data transformations before any rule was fired. Since this would be bothersome to the user and inefficient, a forward-chained inference engine was not a viable option for SHARP.

In order to circumvent this constraint, the rules were written in Prolog, a goal-driven backward-chained language. Previous versions of SHARP employed a LISP-based inference engine and the knowledge base had a significantly different design from the current implementation. Although a forward-chained language such as NASA's CLIPS can be used to obtain data in a more efficient manner (e.g., by simulating backward-chaining) [Hayes-Roth 1983], this tends to dilute the advantages of forward-chaining [Winston 1984] as well as increasing the number of rules significantly. For this application, it was therefore preferable to simply use a backward-chained rule language.

In the Prolog knowledge base, each node of the decision tree is represented by a single rule. Figure 4 shows the relationship between a generic decision tree and the Prolog implementation of that tree. In the decision tree in figure 4, A1, A2, and A3 are decisions to be evaluated, such as "IS the telemetry value for channel X out of limit?", or "IS there bad weather at the DSN station?". B, c, and d are other subtrees of the same form as shown in the figure, that are invoked if one of the A1-A3 are true. "Diagnosis1" is the default diagnosis that is concluded if A1-A3 are all false. Although only one binary decision is made at each node, there may be multiple data values that are examined in order to arrive at that decision. Thus A1-A3 each represent one or more goals to be evaluated. These goals may include calls to C functions that will return the values of telemetry, predicts, or SOE information,

or ask the user for a data value.

To invoke the Prolog rules in figure 4, the goal  $a(X)$  is called (where X is an unbound variable). If A1, A2, or A3 is true, then  $b(X)$ ,  $c(X)$ , or  $d(X)$  is called respectively, passing X on to be instantiated by one of the rule subtrees. When a diagnosis is determined it is unified with X, making it accessible by the top level rule that originally called  $a(X)$ .

The Prolog knowledge base (KB) is embedded in a C program. Whenever a complete set of telemetry values is received from the GDS (approximately every 15 seconds, under ideal conditions), the main C program stores the values in global C structures for later retrieval. Then C invokes the Prolog KB by calling an interface function that calls the top level Prolog goal. The rules are evaluated in the manner described above. If a data value is needed in order to make a decision and the value is available from an on-line data source, a C function is called directly from the Prolog rule to retrieve the required value. If a decision requires a value that is not available on-line, the rule calls a C function that consults the user for the answer by popping up a query window. At that point all telemetry processing and KB inferencing is suspended until the user responds with an answer. When the user inputs an answer, the C function that popped up the query window will return the answer to the Prolog rule that called it, and the rule will then continue on with its evaluation of the current decision node. Note that the suspension of all processing is achieved by virtue of the fact that control will not return to the Prolog rule until the C query function has terminated, and the main C function cannot continue its execution until the Prolog KB has terminated execution.

Each time a decision tree node is visited, the identifier of the rule associated with that node is recorded in the KB. If the rule eventually fails, the identifier is modified to indicate that the node is false. Upon completion of inferencing the user can request an explanation of the steps taken in the diagnosis process. In response to this request, the list of recorded nodes from the most recent diagnosis is translated into English and output to the user in the order in which the nodes were visited.

Occasionally the rules will require that some action be taken in order to attempt to fix a problem that has been diagnosed by the rules, and the result of the fix must be evaluated to determine if any other action is required. For example, if it has been determined that it is raining at the DSN station, then the user must request the station to turn on a blower to evaporate the moisture at the antenna. If the problem is fixed by the action then the KB terminates inferencing and control

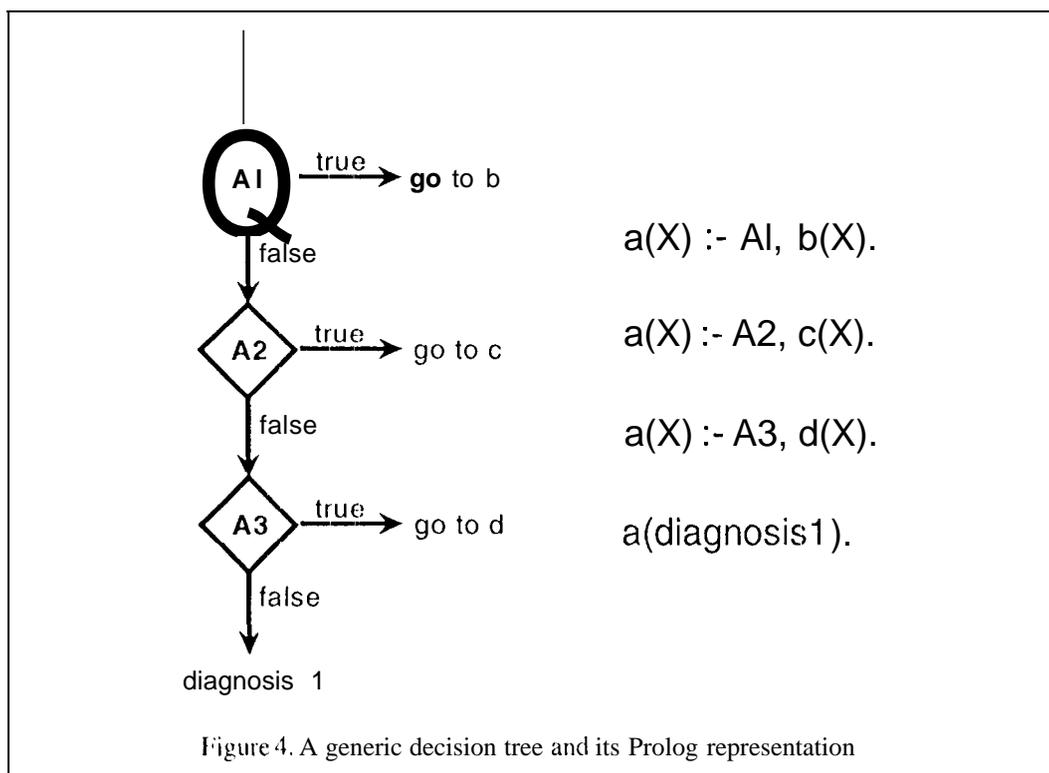
is returned to the main C function. If the problem is still not fixed by the action, then a different corrective action must be attempted, or the KB will determine that the problem cannot be currently fixed. The only way for the KB to determine whether the problem has been fixed is by examining the latest telemetry *after the action has been performed and the effects of the action are visible in telemetry*. Thus while KB processing is suspended pending the execution of the action, the telemetry must continue to flow through the system, without being evaluated, until the effects of the action (if any) show up in telemetry.

When this type of action must be performed, the Prolog rule first suspends KB inferencing by adding (asserting) two facts to the KB, one to indicate that the KB is in a suspended state, and the other to specify the goal to be called upon resumption of processing. Next a C function is called which pops up a window with a message asking the user to perform the required action. Unlike the C function for simple user-queries, the action-request function spawns a new process to pop up the window, which executes independently from the other SHARP processes. The action-request function can then terminate and return control to the calling Prolog rule while the pop-up action-request window is still displayed and waiting for user input. The Prolog rule then completes execution and control returns to the main C function. When the action has been performed and enough time has

passed so that its effect will be visible in telemetry, the user clicks on the "OK" button in the action-request window, signifying that the KB can resume inferencing.

While the KB is in the suspended state, the main C function resumes normal processing, reading telemetry from the GDS and calling Prolog whenever a complete telemetry set is received. It is important that KB inferencing is not restarted until the state of the Telecom system is changed by the action that has been requested. When the top-level Prolog rule is called while in the KB suspended state (as determined by the previously asserted fact) the rule first calls a C function to see if the user has responded to the action request. If the user has responded, indicating that the action has been performed, then the the previously asserted goal (to be executed upon KB resumption) is retrieved from the KB. The goal is then called, which has the effect of returning inferencing to the point where execution had left off just prior to the suspension of the KB. At that point the current telemetry values should reflect the effects of the requested action, so the KB can continue its diagnosis using the latest available data.

If the user has not yet responded to the action-request window, then the top-level rule does nothing further and returns control to the main C function, This causes the latest set of telemetry to be ignored by the KB, because it does not yet show the effects of the action to be performed, and thus does not reflect the



state of the Telecom system that the rules expect to exist upon resumption of inferencing.

### **Application Use and Payoff**

SHARP has been deployed for almost one year, resulting in a variety of operational benefits. SHARP has enabled faster response time in the detection and diagnosis of Telecom anomalies, due to two factors. First, since SHARP automates the knowledge of an expert Telecom analyst, the Telecom expert is no longer needed for problems that are within SHARP's domain, eliminating the time needed to contact the expert and apprise him of the situation. Secondly, the actual time required to detect the anomaly and then to diagnose the cause of the problem and recommend how to fix it are much less than the time required by any human operator.

Faster response time results in increased safety for the spacecraft, since any interruption in communication with the spacecraft could allow serious spacecraft anomalies to go undetected and uncorrected, possibly leading to the loss of the spacecraft, as occurred in the case of the Mars Observer. Also, even in the absence of serious spacecraft anomalies, faster response time means that any downtime of the Telecom link is minimized, thus minimizing the loss of both science and engineering data transmitted from the spacecraft.

SHARP also enables reduction in operational staff size to take place with minimal impact. Past interplanetary missions have staffed the function now carried out by SHARP with a full-time Telecom expert, at a cost of approximately \$100,000 per year for each mission. More recent missions have been under significant pressure to reduce costs, causing them to eliminate a variety of previously funded positions, including dedicated Telecom support. SHARP enables mission controllers to derive on-line consultation from SHARP for all but the most unusual Telecom problems, for which experts are called in on an ad-hoc basis. For these types of problems, SHARP assists experienced analysts with reminders of what parameters are of importance in a particular analysis, and what actions are appropriate in order to solve a given problem.

Also, since SHARP automates Telecom anomaly detection, routine monitoring can be performed by operators whose primary expertise is not in Telecom. This means that a single person can monitor both Telecom and other non-Telecom subsystems, al-

lowing additional reductions in staff size.

Other anticipated benefits of SHARP include training inexperienced analysts by taking them step by step through the diagnosis process, via the knowledge base explanations. A final benefit will be the preservation of Telecom expertise in the SHARP knowledge base, after key Telecom experts have retired or moved on to other missions.

### **Application Development and Deployment**

Development proceeded in two phases. The initial implementation of the system, as discussed in the Application Description section of this paper, involved two full-time developers for one year. One developer was responsible for the knowledge base and the other was responsible for the implementation of the GUI and database. A third developer participated in the task for three months, and was responsible for the telemetry communication interface. Requirements for the knowledge base were obtained from domain experts in the form of decision-tree flow charts. These served as the only formal requirements for the implementation. The system evolved around the knowledge base requirements, as designed by the developers in support of the decision tree implementation. Regular feedback from the domain experts was obtained during informal meetings that were scheduled on an as-needed basis. During these meetings, the user-interface and knowledge base served as the primary means for communicating system capabilities.

Subsequent to the completion of this development phase, SHARP was installed in the Mission Control Center and several weeks of user training ensued. User training was informal, conducted on a one-on-one basis, with one of the developers training a single operator by demonstrating the system, answering questions, and then observing operational use, offering suggestions, and answering additional questions. This was done in order to assure minimal disruption to operational schedules.

One week after the training process was completed, the Mars Observer orbit insertion was scheduled to take place. This was assumed to be an ideal opportunity to evaluate SHARP in a complex operational scenario. However, all communication with the spacecraft was inexplicably lost prior to this maneuver, and the opportunity for evaluation disappeared as well.

As a result, the need for a second development phase materialized, in which SHARP was ported to the Galileo mission to Jupiter. This second devel-

opment phase had a duration of four months, and involved two developers. During the second phase, development activities included expanding the data parameter set (to accommodate additional data that had not been available in Mars Observer telemetry but that enabled more complete diagnosis) and modifying the rule base (to utilize the new parameters and adapt the original decision tree to the unique configuration of Galileo's telecommunication subsystem). Subsequently, the system was again deployed, much in the same way as it had been for Mars Observer.

## Maintenance

The need for maintenance of the SHARP knowledge base, similar to other knowledge bases that we have developed, has been most intense immediately following delivery, as actual use in operations enables analysts to detect problems that were not evident during review of the decision trees or during pre-delivery testing. Most of these changes involve additions to the knowledge base based on the detection of omissions as opposed to implementation errors, though there are some of these as well. Subsequent to the resolution of these initial issues, the need for maintenance tends to decrease exponentially dropping to several hours of maintenance per month over a period of six months.

SHARP has followed this general trend, and has required little or no active involvement from the developers in recent months. The knowledge base is very stable and the reasoning inherent in spacecraft telecommunications is generic in nature. As a result, maintenance responsibility will transfer to the end-users in the near future. This will require a period of consultation, during which the developers are on hand to provide guidance and answer questions, and will eventually enable the end-users to customize the SHARP application to other missions.

The knowledge base, written in Prolog, is in a form that is practical for the end-user to maintain given adequate training from the development team. This is possible in part because the structure of the knowledge base is such that there is a

one-to-one correspondence between rules in the knowledge base and nodes in the decision tree, making it easier to find where a particular node is represented in the rules, and where new rules should be inserted if new nodes are added to the decision tree. We have found that an end-user with some programming experience can, after approximately 3 weeks of fairly frequent though not necessarily time-consuming

consultation, make reasonably independent contributions to the knowledge base. After this amount of time, an end-user would not necessarily be considered a Prolog programmer, but would be sufficiently familiar with the subset of syntax required to update the knowledge base with confidence.

To date, there is no mechanism for automated update of the knowledge base, but we are implementing tools to facilitate modification or addition of rules by automatically verifying them for consistency with previously existing portions of the knowledge base. This will enable a knowledge base to be successfully maintained as a result of successive efforts by several different individuals. This is considered to be an important feature in the operations environment, where it is not common for a single analyst to have dedicated responsibility for a task over the entire duration of a mission.

## Summary

SHARP automates the knowledge of an expert in the domain of spacecraft telecommunications. The knowledge base uses backward chaining to detect and diagnose Telecom anomalies by examining telemetry and other on-line data, and by obtaining information from the user through an interactive consultation mode. This automated analysis provides faster diagnosis and correction of anomalies than has been possible in the past. The knowledge base is adaptable to multiple spacecraft missions by virtue of the fact that the domain knowledge is largely generic in nature and the Prolog decision tree structure of the rules allows the knowledge base to be easily modified and expanded. SHARP makes workforce reductions less painful by reducing the dependence on Telecom domain experts, and by allowing mission analysts from other subsystem domains (e. g., attitude control, power) to perform Telecom monitoring and analysis.

## Acknowledgment

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aero-

nautics and Space Administration. The authors appreciate support from the NASA Office of Advanced Concepts and Technology and JPL's Multimission Operations Support Office.

## References

Forgy, C. 1982. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence* 19:17-37.

Hayes-Roth, F. 1983. *Building Expert Systems*. Reading, Mass.: Addison-Wesley. 172-215.

Martin, P (cd.); Atkinson, D.; James, M.; Lawson, D; and Porta, H. 1990. Spacecraft Health Automated Reasoning Prototype (SHARP): A Report on SHARP and the Voyager Neptune Encounter. Publication 90-21. Jet Propulsion Laboratory, Pasadena, CA.

Schwuttke, U. M.; Quan, A. G.; Angelino, R.; et al. 1991. MARVEL: A distributed Real-time Monitoring and Analysis Application. In Innovative Applications of Artificial Intelligence 4,89-106. Menlo Park, Calif.: AAAI Press.

Schwuttke, U. M.; Veregge J. R.; and Quan, A.G. 1994. Cooperating Expert Systems for the Next Generation of Real-time Monitoring Applications. In Proceedings of the Second International Conference on Expert Systems for Development, 210-215. Los Alamitos, Calif.: IEEE Computer Society Press.

Shortliffe, E. H.; and Buchanan, B.G. 1976. *Computer-based medical consultation: MYCIN*. New York: American Elsevier.

Winston, P. 1984. *Artificial Intelligence*. Reading Mass: Addison-Wesley. 152-153.