N90-27336

SDI SATELLITE AUTONOMY USING AI AND ADA

Harvey E. Fiala

SBI Software Engineering
Rockwell International
Strategic Defense Center
2800 Westminster Blvd.
P. O. Box 3089, MC/EA20
Seal Beach, CA 90740-7644

Telephone: (213) 797-4496

ABSTRACT

This paper describes the use of artificial intelligence (AI) and the programming language Ada to help a satellite recover from selected failures that could lead to mission failure. An unmanned satellite will have a separate AI subsystem running in parallel with the normal satellite subsystems. A satellite monitoring subsystem (SMS), under the control of a blackboard system, will continuously monitor selected satellite subsystems to become alert to any actual or potential problems. In the case of loss of communications with the earth or the home base, the satellite will go into a SURVIVAL mode to reestablish communications with the earth. The use of an AI subsystem in this manner would have avoided the tragic loss of the two recent Soviet probes that were sent to investigate the planet Mars and its moons.

The blackboard system works in conjunction with an SMS and a reconfiguration control subsystem (RCS). It can be shown to be an effective way for one central control subsystem to monitor and coordinate the activities and loads of many interacting subsystems that may or may not contain redundant and/or fault-tolerant elements. The blackboard system will be coded in Ada using tools such as the ABLE development system and the Ada Production system.

INDEX TERMS—Ada, autonomy, blackboard, expert system, frame, global data base, inference engine, knowledge base, and rule base.

INTRODUCTION

Two Soviet probes, Phobos 1 and Phobos 2, were recently sent to investigate the planet Mars and its moons, but were both lost. A ground controller had sent an unverified command that caused loss of communication to the earth. Command verification was not possible because the Soviet's ground control computer, responsible for validating uplink command sequences was down. As a result, no further uplinked ground commands could be received, and the batteries went dead after the spacecraft lost solar panel orientation. This was truly a profound loss to space science. The Spacecraft Autonomy Group criticized the Soviets for allowing unverified uplinks, and for having an on-board computer that so easily accepted such single-transmission command sequences. The United States' Voyager spacecraft will only accept a command sequence that has been repeated three times. An autonomous system design that included an AI subsystem, as described in this paper, would have enabled the Soviet probes to recover from their tragic circumstances.

Examples of expert systems that provide some degree of autonomy for satellites include (1) the Expert System for Satellite Orbit Control (ESSOC) [Reference 2], which provides autonomous processing for satellite maneuvering operations, (2) the Autonomous Satellite Control [Reference 3], which allows a satellite to operate on its own for up to 30 days; and (3) Spacecraft Control Resolution Expert System (SCARES) [Reference 4], which handles anomalies in a satellite's attitude control system. Other spacecraft with varying degrees of autonomy are reported in [Reference 11].

The GIOTTO spacecraft, which successfully encountered Halley's comet in March 1986, had a number of autonomous facilities on board. These ranged from the simple switching of heaters, to the autonomous reconfiguration of on-board subsystems, extending to the full autonomous recovery of contact with earth [Reference 9].

The Indian Remote Sensing Satellite (IRS), launched in March 1988 into a polar sun-synchronous orbit, achieves a high degree of autonomy by possessing many fault-tolerant features, including automatic reconfiguration logic for the attitude control system [Reference 5].

The Infrared Space Observatory (ISO), planned for launch into a 24-hour orbit in 1993, will require considerable autonomous operation and reconfiguration capability [Reference 12]. Autonomy features will permit recovery of the satellite in good health and enable a quick restart of scientific operations after a period of up to 3 days without earth contact.

Rockwell International has prepared a final report to NASA titled, "Research On Advanced Engineering Software for In-Space Assembly and the Manned Mars Spacecraft" [Reference 1]. This report identifies a strong need for advanced engineering software to support spacecraft autonomy and subsystem health maintenance. It identifies Intelligent Communicating Agents (ICA), which is a form of intelligent distributed software processing, as one example of Advanced Engineering Software directly applicable to future NASA space missions and objectives.

Graceful degradation of overall spacecraft performance takes place as various subsystems fail by using prioritized loading charts contained in the blackboard systems knowledge base. For example, if a failure of a power supply takes place and no spare power supply exists to replace it, then the various loads on any remaining power supplies are either turned off, reduced, or switched to a duty cycle tolerable to the remaining power supply. According to a prioritized table, the least critical functions are either removed or placed on a low duty cycle consistent with minimum mission objectives.

Power supply degradation can occur due to component failures, aging, distance of the solar panels from the sun, half-life of radioisotope power sources, and other potential reasons. Designing independently redundant subsystems could result in an overload of the power system during degraded power system performance. However, if all subsystem reconfiguration is coordinated through a central subsystem, then various subsystem loads can be reconfigured to accommodate a reduced available power, while still meeting overall mission objectives. Under conditions of reduced power, data collection is reduced resulting in less required transmitter power. The result is a graceful degradation of system operation. It is shown here that a blackboard system is a good way to accomplish this reconfiguration management.

AI BLACKBOARD SUBSYSTEM

Figure 1 in the Appendix gives a high-level block diagram for a blackboard system for an autonomous satellite. Blackboard systems provide a mechanism to implement cooperation between a collection of expert systems or knowledge sources. Blackboard systems consist of an explicit global data base (called the blackboard) and knowledge sources that effect and react to changes on the blackboard. Differences among blackboard systems involve mainly control algorithms and mechanisms for determining when knowledge sources should be executed.

The blackboard system works in conjunction with an SMS and an RCS. In the event that a mission-threatening condition is taking place or has occurred, then a corrective action is taken by the RCS. The corrective action is based on various redundant and fault-tolerant features that are integrated into the satellite as part of the overall autonomous design.

The blackboard system is a logical way for one central control subsystem to monitor and coordinate the activities and loads of many interacting subsystems that may or may not contain redundant and/or fault-tolerant elements.

A system or satellite reconfiguration might go as follows: Suddenly the satellite power drops to the 90-percent level. The SMS detects this power drop and sets the relevant flag on the blackboard. The inference engine (scheduler), which monitors the blackboard, is alerted and examines the knowledge sources in the knowledge base and finds a rule that reconfigures the satellite according to priority loading Table 1 in the Appendix. The RCS, alerted from the blackboard, then studies priority Table 1 from the knowledge base and the current system configuration from the blackboard and identifies any required changes. It studies the frame data for the affected subsystems from the blackboard to determine any constraints. In case of a conflict between two units that should not be on together, or between two units that must be on together, an overall priority table in the knowledge base is consulted. The RCS then schedules or reconfigures the affected subsystems according to the new priority table. It then updates the system configuration data on the blackboard.

KNOWLEDGE BASE (LONG-TERM MEMORY)

The knowledge base contains the rule base and the various priority tables.

Examples of two rule templates of the type used in the knowledge base are:

RULE_SPARE_UNIT: If one UNIT fails, and at least one spare UNIT does exist; then switch out the failed UNIT, and switch in the spare UNIT.

RULE_NO_SPARE_UNIT: If one UNIT fails, and a spare UNIT does not exist, and at least one UNIT is still operating, and a prioritized loading table does exist;
then reduce the load on the operating UNIT(s) according to the prioritized loading table for the UNIT.

A partial rule base for autonomous SDI satellite subsystem reconfiguration is given in Table 1 in the Appendix.

Load types are classified into types A through D as follows:

Type A: Loads that must be run at 100-percent power and 100-percent duty cycle (for example, the executive or an IMU). An estimated 10 percent of the loads falls into this category.

Type B: Loads that must be run at 100-percent power when they are on, but can be run at a duty cycle of less than 100 percent (for example, if an output is not needed all the time, as in a radio receiver that needs to be powered on only during reception, a transmitter that needs to be powered on only during transmission, an attitude control system that needs to be powered on only during attitude control, and an on-board signal processor that periodically processes data prior to transmission to the earth). An estimated 75 percent of the loads falls into this category.

Type C: Loads that can run at less than full power, but must run continuously (for example, volatile memory, timing sources). An estimated 5 percent of the loads falls into this category.

Type D: Loads that can run at less than full power and at less than 100-percent duty cycle (certain heaters, coolers, etc). An estimated 10 percent of the loads falls into this category.

An example of a simplified prioritized loading table is given in Table 2 in the Appendix.

GLOBAL DATA BASE (THE BLACKBOARD)

The global data base, which is the blackboard, contains the status of all subsystems, frame data on all the subsystems, and the flags for the various subsystems to communicate with each other. Frames are used to collect all required data on each subsystem. In this application, executable procedures are not attached to each slot in the frame, as is the case in the typical blackboard system.

An example of a frame for one of the subsystems is given in Table 3 in the Appendix.

SATELLITE MONITORING SUBSYSTEM (SMS)

The SMS monitors excursions beyond temperature and voltage limits, attitude deadbands, and spin rate limits; monitors selected status flags, time since contact with the satellite's home base, duration of thrust pulses, verification of command sequences; and other mission-critical functions. In the event that a mission-threatening condition is taking place or has occurred, then the SMS sets the appropriate flag on the system blackboard to alert the Inference Engine and the RCS.

RECONFIGURATION CONTROL SUBSYSTEM (RCS)

The RCS monitors the blackboard for changes in the system status. It checks the priority tables in the knowledge base, and it has the ability to reconfigure each of the reconfigurable subsystems.

Reconfiguration of a subsystem can be achieved by turning it off, by turning it off and bypassing it, by changing the duty cycle of the power supplied to it, or by lowering the power supplied to it. After the RCS reconfigures a subsystem, it updates the subsystem's status on the system blackboard.

Table 4 in the Appendix contains a listing of generic satellite sussystems with what is probably their most likely load type.

SURVIVAL MODE

If communication is not reestablished after a certain time, or if the battery charge falls below a prespecified limit, then the SURVIVAL mode is engaged. Power is removed from all possible subsystems and, at a predetermined time, the home base, on or near the earth, will start to transmit a very strong signal, according to a prearranged plan. Or as an alternative, the satellite will go into a star- and sun-tracking mode. At the predetermined time, the satellite will then do an attitude scan with its receiver to lock onto the strong signal from the home base.

The home base will alternate transmission of the strong signal with listening with its receiver until communication with the home base has been reestablished.

IMPLEMENTATION IN ADA

The Ada language has been mandated as the official software language by the DOD, and NASA is now rapidly moving in that same direction for new software programs. There is an advantage to implementing AI programs in Ada from the standpoint of standardization, life cycle maintenance, and customer acceptance.

The blackboard system would be developed first on the Generic Blackboard System (GBB) and then recoded in Ada. The rule base portion of the system would be converted into Ada code using the Ada Production System (APS) [Reference 7]. The APS is a development tool that was developed at Rockwell International and used to develop knowledge-based applications written in Ada. To the author's knowledge, there currently are no commercially available expert system shells that produce executable Ada code. Benchmark testing of Ada code produced by the APS shows that the code executes at approximately 75 percent of the speed of OPS83 code. OPS83 code is based on the C language and is considered to be the fastest production system in use.

Examples of AI systems that have been coded in Ada include the LATEST expert system, the ABLE blackboard system, and the Embedded Rule-Based System (ERS).

LATEST is a very successful rule-based expert system coded in Ada. It gives the reason for a hold or an abort of a Shuttle launch within 3 seconds, where this process normally takes experts several hours of analysis. Knowledge base rules were generated from example sets by a process called rule-induction using the RuleMaster Expert System development tool. It kept the software conventional by avoiding an inference engine. Quoting from Reference 14, "GHC had already made compiling rules for real-time execution possible by developing RadAda to translate RuleMaster interpretive code into Ada source code."

The ABLE system consists of a development system (compiler or constructor) and a library. To create the blackboard system using ABLE, either Erasmus and the ABLE compiler, or the ABLE Constructor would be used. Paraphrasing from Reference 13, "The object of the ABLE compiler is to accept Erasmus source code and produce an equivalent Ada program. The object of the ABLE constructor is to provide the application developer with a graphic interface for defining the structure of a blackboard application. The constructor then performs most of the code generation automatically, producing Ada source code that the developer may fine tune at will."

Quoting from Reference 13, "The ABLE development system may not have been able to fully resolve certain data typing questions, nor may it know details of communicating with other software or devices, so the user may manually extend and/or optimize the code at this point. Code produced will already contain references to all necessary ABLE library units, as well as pertinent sublibrary units, but the user may of course add references when extending the code."

The ERS was successfully recoded in Ada. Quoting from Reference 6, "The project evolved into a major redesign of ERS that exploits Ada's facilities for data abstraction and object-oriented development. The resulting Ada implementation has all of the functionality of earlier versions of ERS (with hooks for many additional features), maintains upward compatibility with existing rule bases, is significantly more efficient than previous versions, and is of higher overall quality by any software engineering standards. Most important, the project demonstrates, convincingly, Ada's suitability and utility for developing knowledge-based systems and embedded AI applications in general."

Ford Lisp-Ada Connection (FLAC) [Reference 8] is a tool designed to support direct entry of knowledge by experts into a Lisp machine environment to help develop expert systems. Paraphrasing from Reference 8, "The knowledge is then downloaded to an inference engine that has been implemented in the Ada programming language. FLAC consists of two subsystems, the Knowledge Editor Graphics System (KEGS) and the Ford Ada Inference Engine (FAIE). The inference engine is written in Ada. It supports both forward and backward chaining modes of inference. FLAC is an application independent system that is generic in the sense that any knowledge that can be represented in rule format can be entered using KEGS, and any Ada program can embed FAIE for expert system capabilities."

That expert system shells, blackboard systems, and other rule-based systems can be successfully coded in Ada can be seen from References 6, 7, 8, 13, and 14, as described above. Rockwell has a very broad set of Ada capabilities [Reference 10], has produced numerous avionics systems coded in Ada, and a blackboard system for an unmanned autonomous satellite is well within its present capabilities.

CONCLUSIONS

This paper has described the architecture for a blackboard system that will coordinate between various satellite subsystems and expert systems to result in the high degree of autonomy required for important defense systems. Sufficient evidence currently exists to demonstrate that AI subsystems can be coded in Ada and embedded in conventional Ada code. In the interests of justifying expenditure of natural resources on important space missions, it should be considered a requirement that each satellite contain an autonomous system that is able to reestablish both vehicle attitude and communications in the event that one or both were lost. Agreeing on certain minimum, internationally developed software standards for satellites and spacecraft can help lead to international cooperation on large-scale missions, such as the Manned Mars Mission.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Don DeLaquil for his critical review of this paper and for his helpful suggestions and comments.

AUTHOR BACKGROUND

Harvey Fiala has an MS in electrical engineering from the California Institute of Technology, a Certificate in AI from UCLA, and a BS in electrical engineering from North Dakota State University. He is a Registered Professional Engineer in the State of California and has been a software engineer for 19 years at Rockwell International. He is a member of IEEE, American Association for Artificial Intelligence (AAAI), the Computer Society, the Robotics and Automation Society, and the Planetary Society.

Mr. Fiala is currently responsible for the development of seeker software for the Space-Based Interceptor (SBI) at Rockwell's Strategic Defense Center in California. Prior accomplishments at Rockwell include the development of a two-way Laser Communication System, an Electronic Warfare Jammer, and a multiband pulse sorter. He performed software verification testing on the Shuttle program. He holds patents in the field of laser communications, PRR signal sorters, and DC-to-DC converters.

Prior experience includes project engineer on an infrared seeker-guided missile and experiment engineer on the Surveyor Spacecraft at Hughes Aircraft. He worked on the stable platform for the inertial guidance system for the Polaris missile at General Electric and on the Talos missile guidance system at Bendix Guided Missiles.

Mr. Fiala's primary interest is in the area of AI. He has coded prototype expert systems in the languages of LISP, Prolog, OPS 83, M1, and Copernicus. He has written and presented, at various national conferences, several papers in the field of AI.

REFERENCES

[1] DeLaquil, D., T. Reid, J. Hoey, and J. Irwin. Research On Advanced Engineering Software For In-Space Assembly and the Manned Mars Spacecraft. Final Report to the Office of Exploration, NASA, Ames Research Center, Moffett Field, California, the Rockwell International Corporation, Space Transportation Systems Division, Expert Systems Application Group, Downey, California (June 30, 1989).

[2] Fiala, Harvey E. Aerospace Applications of Artificial Intelligence—A Survey. Aerospace Applications of Artificial Intelligence Conference (AAAIC 88), Dayton, Ohio (Oct. 25-27, 1988).

[3] Fiala, Harvey E. Artificial Intelligence Applications in Space and SDI—A Survey. The 4th Conference on AI for Space Applications, NASA MSFC, Huntsville, Alabama (Nov. 15-16, 1988).

[4] Fiala, Harvey E. Artificial Intelligence Applications to Real-World Problems—A Survey. SES III, Rockwell International Third Annual Software Engineering Symposium, Richardson, Texas (Oct. 2-4, 1989).

[5] Goel, P.S., S. Murugesan, N.K. Malik, and K.S. Chandra. Autonomous Safe Mode Operations of the Attitude Control System of Indian (IRS and INSAT-II) Satellites to Prevent Catastrophe, Present and Future Capabilities. Sponsored by the International Federation of Automatic Control (IFAC), organized by JPL on behalf of the American Automatic Control Council (AACC), JPL, Pasadena, California (Sept. 13-15, 1988).

[6] Hirshfield, S.H., and T.B. Slack. ERS: An Expert System Shell Designed and Implemented in Ada. AIDA-88, Fourth Annual Conference on Artificial Intelligence and Ada, Herndon, Virginia (Oct. 14-15, 1987).

[7] Irwin, J.A. Symbolic Processing in Ada. SES III, Rockwell International Third Annual Software Engineering Symposium, Richardson, Texas (Oct. 2-4, 1989).

[8] Jaworski, A., PhD., D. LaVallee, and D. Zock. A LISP-Ada Connection for Expert System Development. AIDA-87, Third Annual Conference on Artificial Intelligence and Ada, Herndon, Virginia (Oct. 14-15, 1987).

[9] Nye, H.R. Onboard Autonomy Considerations for the GIOTTO Mission to Halley's Comet, Spacecraft Autonomy: Present and Future Capabilities. Sponsored by the IFAC, organized by JPL on behalf of the AACC, JPL, Pasadena, California (Sept. 13-15, 1988).

[10] Post, J.V. Ada Capabilities. Rockwell International, Space Transportation System Division, distributed at the Seventh National Conference on Ada Technology, Atlantic City, New Jersey (Mar. 13-16, 1989).

[11] Post, J.V. International Federation of Automatic Control Workshop in Spacecraft Autonomy, Present and Future Capabilities. Trip Report, Jet Propulsion Laboratory (Sept. 13-15, 1988).

[12] Robson, A. and W. Van Leeuwen. Autonomous Operation and Reconfiguration Features of ISO and Their Impact on the Ground Segment. Sponsored by the IFAC, organized by JPL on behalf of the AACC, JPL, Pasadena, California (Sept. 13-15, 1988).

[13] Stockman, S.P. ABLE: An Ada-Based Blackboard System, AID3-88. Fourth Annual Conference on Artificial Intelligence and Ada, Herndon, Virginia (Oct. 14-15, 1987).

[14] Wood, K., L.G. Hanely, B. Lawson, C. Randall, R. Turner, and Wang Chi-Wei. Shuttle Failure Detection, (using LATEST, a Rule-Based Expert System in Ada). Aerospace American (July 1989).

APPENDIX

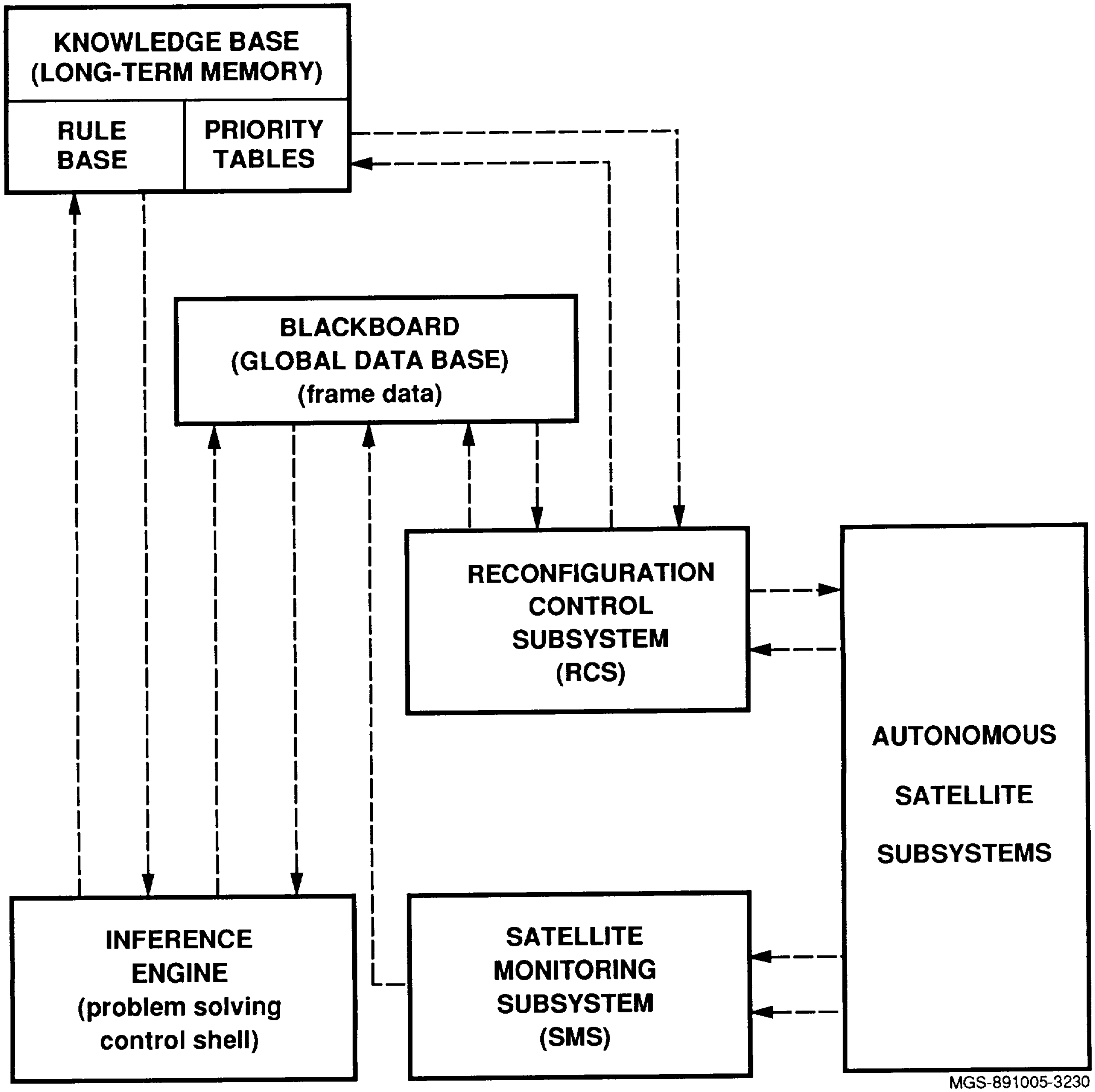

Figure 1. Blackboard System for Autonomous SDI Satellite

Table 1. Partial Rule Base for Autonomous SDI Satellite Subsystem Reconfiguration

RULE_SPARE_HEATER: If one HEATER fails,
and at least one spare HEATER does exist;
then switch out the failed HEATER,
and switch in the spare HEATER.

RULE_NO_SPARE_HEATER: If one HEATER fails,
and a spare HEATER does not exist,
and at least one HEATER is still operating,
and a prioritized loading table does exist;
then reduce the load on the operating HEATER(s)
according to the prioritized loading
table for the HEATER.

RULE_SOLAR_PANEL: If one SOLAR_PANEL fails,
and at least one spare SOLAR_PANEL does exist;
then switch out the failed SOLAR_PANEL,
and switch in the spare SOLAR_PANEL.

RULE_NO_SPARE_SOLAR_PANEL: If one SOLAR_PANEL fails,
and a spare SOLAR_PANEL does not exist,
and at least one SOLAR_PANEL is still operating,
and a prioritized loading table does exist;
then reduce the load on the operating SOLAR_PANEL(s)
according to the prioritized loading table for
the SOLAR_PANEL.

RULE_SPARE_BATTERY: If one BATTERY fails,
and at least one spare BATTERY does exist;
then switch out the failed BATTERY,
and switch in the spare BATTERY.

RULE_NO_SPARE_BATTERY: If one BATTERY fails,
and a spare BATTERY does not exist,
and at least one BATTERY is still operating,
and a prioritized loading table does exist;
then reduce the load on the operating BATTERY(s)
according to the prioritized loading
table for the BATTERY.

RULE_LOST_CONTACT: If LOST_CONTACT_FLAG is set
then initiate the SURVIVAL mode.

RULE_CONTACT_REESTABLISHED: If
CONTACT_REESTABLISHED_FLAG is set
then resume the NORMAL mode.

Table 2. Electrical Power Loading Reconfiguration Priority Table. When the Available Power Drops by 10 Percent, the Loads Are Reconfigured According to the Table

Load Type and No.	**Power Supply Output Level (%)**									
	100	**90**	**80**	**70**	**60**	**50**	**40**	**30**	**20**	**10**
A1	10	10	10	10	10	10	10	10	10	10
B1	20	15a*	10a	10a	10a	10a	5a	5a	5a	0
B2	20	15a	10a	10a	10a	10a	5a	5a	5a	0
B3	10	10	10	5a	5a	5a	0	0	0	0
B4	10	10	10	5a	5a	5a	0	0	0	0
D1	30	30	30	30	20	10	10	0	0	0
Total	100	90a	80a	70a	60a	50a	40a	30a	20a	10

Note*: a = average power achieved by lowering the duty cycle. For example, 15a means 15 W average. The duty cycle is lowered.

Table 3. An Example of a Subsystem Frame

Subsystem name:	Laser communication receiver
Constraints:	Must not be on simultaneously with laser transmitter
Power required:	5 W
Load type:	B
Number of spares:	One
Status:	Operational
Previously bypassed:	No

Table 4. Generic Satellite Subsystems and Load Types

Satellite Subsystem	Typical Load Type
Autonomy subsystem	A
Electrical power system	A
Executive	A
IMU	A
ACS	B
Axial thrusters	B
Data processor	B
Divert thrusters	B
GN&C	B
High-power experiments	B
Low-power experiments	B
Receiver	B
Self-test	B
Solar collector	B
Star tracker	B
Tape recorder	B
Clock (timing reference)	C
Volatile memory	C
Heater	D
Transmitter	D