N93-11932

SPACECRAFT ATTITUDE CONTROL USING A SMART CONTROL SYSTEM

Brian Buckley
Interface & Control Systems
1944 South Dairy Road
West Melbourne, Florida 32904

Abstract

Traditionally, spacecraft attitude control has been implemented using control loops written in native code for a space hardened processor. The Naval Research Lab has taken this approach during their development of the Attitude Control Electronics (ACE) package. After the system was developed and delivered, NRL decided to explore alternate technologies to accomplish this same task more efficiently. The approach taken by NRL was to implement the ACE control loops using expert systems technologies. The purpose of this effort was to:

- Research capabilities required of an expert system in processing a classic closed-loop control algorithm.
- Research the development environment required to design and test an embedded expert systems environment.
- Research the complexity of design and development of expert systems versus a conventional approach.
- Test the resulting systems against the flight acceptance test software for both response and accuracy.

Two expert systems were selected to implement the control loops. Criteria used for the selection of the expert systems included that they had to run in both embedded systems and ground based environments. Using two different expert systems allowed a comparison of the real-time capabilities, inferencing capabilities, and the ground-based development environment.

Louis Wheatcraft
Barrios Technology, Inc.
1331 Gemini Ave
Houston, Texas 77058

The two expert systems chosen for the evaluation were Spacecraft Command Language (SCL), and NEXPERT Object. SCL is a smart control system produced for the Naval Research Lab by Interface and Control Systems (ICS). SCL was developed to be used for real-time command, control, and monitoring of a new generation of spacecraft. NEXPERT Object is a commercially available product developed by Neuron Data.

Results of the effort were evaluated using the ACE test bed. The ACE test bed had been developed and used to test the original flight hardware and software using simulators and flight-like interfaces. The test bed was used for testing the expert systems in a "near-flight" environment.

This paper details the technical approach, the system architecture, the development environments, knowledge base development, and results of this effort.

Introduction

The Naval Research Lab has developed an upper stage used for orbital insertion of satellites. The upper stage is spin stabilized until it reaches the insertion orbit. Once in the desired orbit, the upper stage is spun down and stabilized using momentum wheels and reaction control thrusters. The upper stage then jettisons the spacecraft allowing it to move into its parking orbit.

All aspects of the orbital transfer maneuver are controlled by the Attitude Control Electronics (ACE).

The ACE subsystem is semi-autonomous and can issue thruster commands to maintain the desired attitude. The ACE control loops were developed in the flight processor's native assembly language. The development of the algorithms required years of design, testing and elaborate simulation.

Once the ACE system had been successfully launched, the NRL began exploring alternate technologies for developing the same task. The NRL selected an expert system approach since it is event driven and would allow the ACE system to take advantage of 90's technologies.

The NRL development effort was to focus on the following goals:

- Test expert system technologies to prove productivity could be increased, and the system could be reused for other needs.
- Research the use of expert system technologies to implement real-world closed-loop control algorithms.
- Research the type of development environment that would be required to develop and test an expert system for closed loop control, in contrast to the traditional environment used to develop embedded systems.
- Research the complexity of an expert system design and the difficulty in developing the expert system compared to a traditional approach.
- Compare the performance and the accuracy of the resulting expert system against the proven flight implementation.

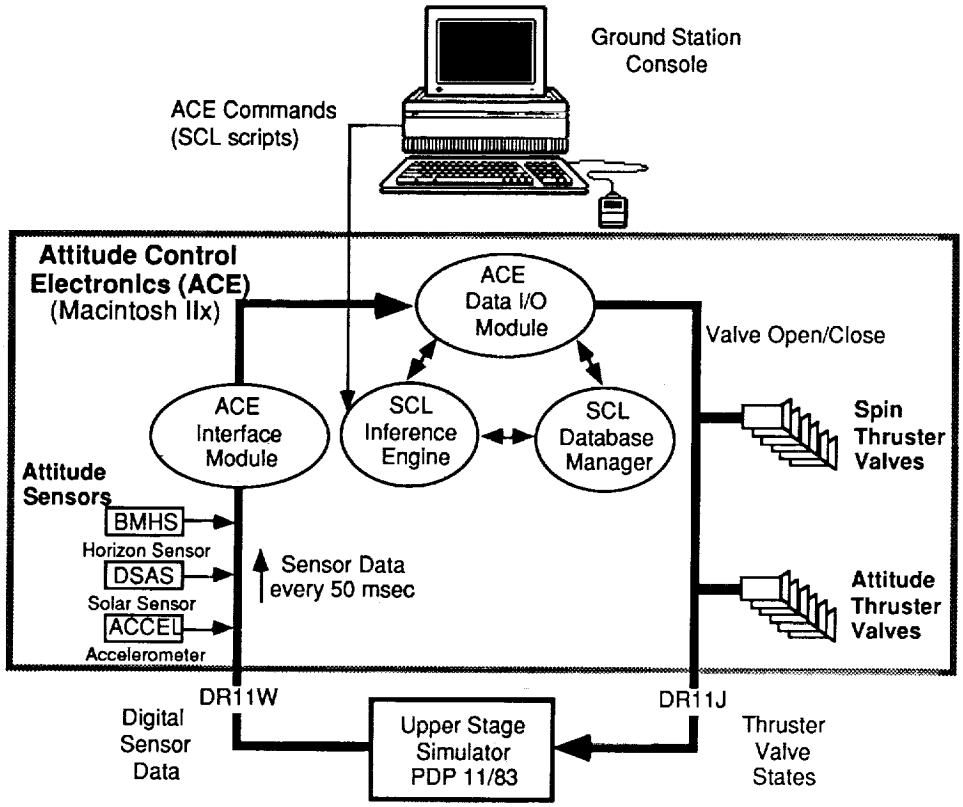
To objectively develop and evaluate the expert system approach, two expert systems were chosen for the development effort. The expert systems that were chosen were required to run in both embedded systems and ground based environments. Both SCL and NEXPERT were chosen since they both were available in an embedded environment. Both systems also have a ground based development environment that is available using an intuitive man-machine interface. The Macintosh II was chosen as the development environment for both expert systems.

Spacecraft Command Language (SCL) is an expert system that was developed for a NRL satellite controller by Interface and Control Systems, Inc. NEXPERT Object is a commercial expert system that was developed by Neuron Data. The two expert systems use radically different approaches to the implementation of the embedded system application. SCL is a total control environment that runs on the embedded processor. The processor, I/O, and operating system specifics are isolated to a small number of low level routines. The SCL system is bound to the operating system specific calls and hardware interfaces using a small amount of "glue" code. The flight algorithms are in the form of scripts and rules and are written in the SCL fifth-generation language. Scripts and rules are compiled on the ground and uploaded to the embedded system where they are interpreted by the SCL real-time executive.

The NEXPERT Object system consists of a library of callable routines that are called from the application code. The application code is written in "C" and makes calls to the NEXPERT library for control of the rule evaluation and inference strategy. The NEXPERT rules are written in a high level language, and are compiled on the ground and interpreted by the on-board target processor.

The NRL contractor that implemented the original ACE flight algorithms was chosen to implement the expert systems in both SCL and NEXPERT. The contractor was familiar with the flight algorithms as well as the flight simulator, and had no contractual ties to either Interface and Control Systems, or Neuron Data.

The ACE flight hardware had been tested against a simulator that gave a three axes model of the spaceborne upper stage. The simulator provided scenarios of normal spacecraft maneuvers as well as variations with anomalies introduced which would require the ACE algorithms to take corrective measures.



SCL ACE System Architecture

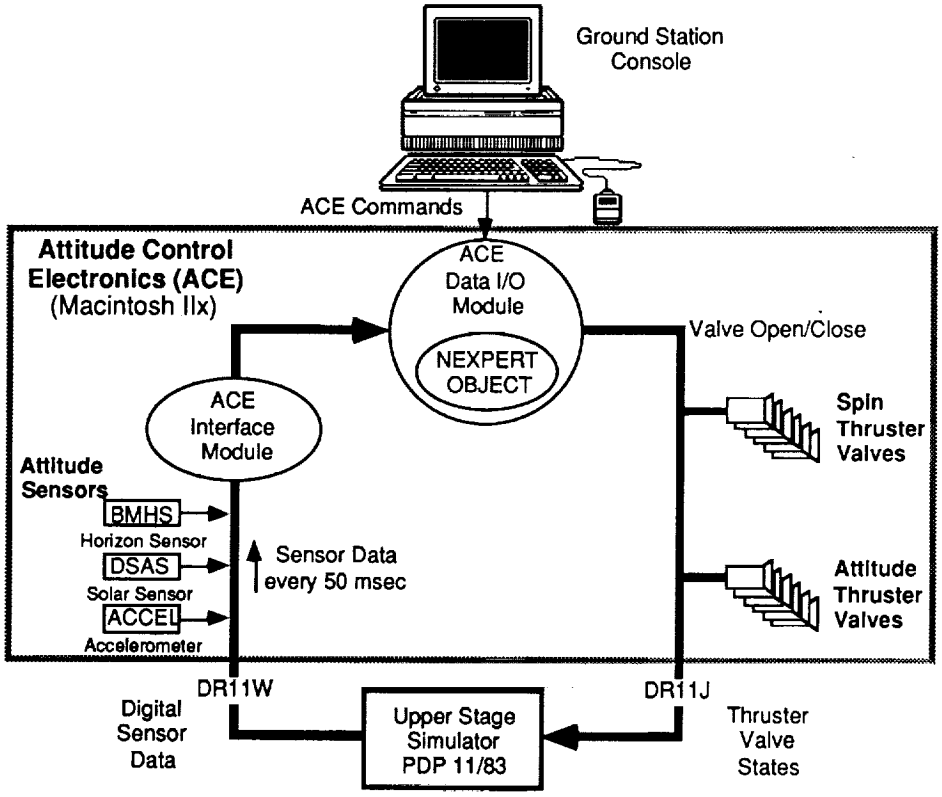
Development Approach

The contractor chose to use the Macintosh II as both the development and run-time environment for the NEXPERT and SCL expert systems. The Macintosh 68000-based machine was roughly equivalent to the flight processor in terms of throughput. The ACE simulator was hosted on a PDP 11/83 and was coupled to the Macintosh using two parallel I/O boards. One board was used to receive sensor data from the simulator, the other board was used to issue commands to initiate thruster maneuvers.

Custom code was developed for both expert systems to support the ACE prototype. The SCL expert system required "glue" code to be written in "C" to couple the bidirectional data between SCL and the I/O boards. The NEXPERT application executive was written in "C". The executive made calls to the NEXPERT library to control the expert system, it also performed tasks that could not easily be implemented using NEXPERT's rules. The NEXPERT data I/O module, which was much the same as that used for SCL, performed all communications with the simulator. Additionally, the I/O module performed timing functions that were not available with NEXPERT.

System Architecture

The ACE simulator emits an 11 word packet of raw sensor data every 10 msecs. A parallel interface was required to ingest the packet, decommutate the sensor data, translate the data into engineering units, and notify the expert system every 10 msecs. The engineering unit form of the sensor data is used by the expert systems for evaluation by rules and scripts. The interface software also stores every fifth packet for attitude filter and nutation calculations.

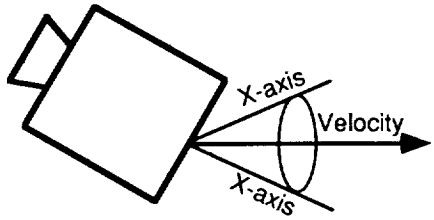


NEXPERT ACE System Architecture

The sensor data is used by the expert systems to perform four closed loop control tasks:

- Spin Rate Control
- Sun Angle Control
- Active Nutation Control
- Active Nutation Filter Calculations

The upper stage does not maintain a constant center of gravity since one or more satellites may be jettisoned. The ACE algorithms must take the changing center of gravity into consideration when making attitude adjustments. The upper stage is normally spin stabilized. Under optimum conditions, the spacecraft spins about its X axis around the velocity vector and both are parallel. Nutation is introduced when the spacecraft begins to wobble. In a simple scenario, one end of the spacecraft begins to wobble as depicted in the drawing to the right. The X axis "cones" about the velocity vector. The cone angle is nominally kept to +/- 0.25° by the ACE algorithms. (The cone angle is discussed later during the evaluation of the systems.) The ACE control loops must determine the appropriate thruster(s) to fire, the exact time of the firing, the duration of the burn, and the number of burns required to correct the nutation.



Upper Stage Nutation

The ACE interface software receives raw sensor data from Digital Solar Aspect Sensors (DSAS) and Body Mounted Horizon Sensors (BMHS). These sensor readings are used to calculate engineering values for the

spin period, sun angle, and active nutation. If the knowledge base determines a corrective action is required, a command is sent to enable the appropriate thruster for some delta time. Corrective actions may be taken to correct either the spin rate, sun angle, or active nutation. All corrections take place based upon commands generated by the ACE algorithms.

Software Development

The accuracy of the thruster maneuvers is critical to correcting the attitude of the spacecraft. The thruster valve open and close times are critical. With precise timing, the number of corrections can be reduced, thus minimizing the amount of reaction control propellant used.

Since NEXPERT had no precision timing capabilities, the functionality had to be provided by writing "C" code in the interface module to implement the capability.

SCL not only provides a language that supports the definition of rules, the language also allows procedural programming using scripts that share a common syntax. The SCL scripts provide a time synchronized execution and are used for the precision timing required for thruster maneuvers.

The ACE prototype also required a method of enabling and disabling various control modes. Again this was handled by SCL scripts, while additional source code was written for NEXPERT to handle this requirement.

Knowledge Base Development

Both expert systems provide a window-based development environment for defining the knowledge base. The NEXPERT system provides a graphical view of the rule heritage, SCL had no graphic representation at the time. The NEXPERT graphics allowed the engineers to view the relationship between database items and rules in the knowledge base. Both systems allow the user to define the knowledge base using a windowed text editor.

The SCL expert system allows the user to mix scripts and rules to form the knowledge base. Scripts may be called by rules using SCL. The SCL scripting feature reduced the complexity of the knowledge base. The english-like scripting capability of SCL allows loops and control structures to be written. These loops and control structures are difficult to construct using rules. Scripting was also used to enable and disable the various control modes. The NEXPERT system required all control algorithms be written as rules and the control functions be implemented by developing additional rules or "C" source code.

The Inference technique of both systems came into play when designing the knowledge base. Both systems are primarily forward chaining expert systems. The NEXPERT inference method employed a prioritized queue for sequencing the execution of rules. The "agenda" allows rules to be dynamically scheduled for execution based on priority. As rules are fired and database items are changed, new rules are merged onto the agenda.

At the time of the ACE project, the SCL inference method used a depth-first strategy. When a database item changes, the rules that reference the item were executed in a prioritized order. If a database item was changed, the inference engine focused on that item and executed rules associated with it before returning to the rules for the previous item. Because of the depth-first method, rules of lower priority could be evaluated before rules of higher priority. This was caused by the system focusing on the last item which changes. Since the ACE prototype was developed, the SCL inference engine has been modified to allow a prioritized queue of rules to be evaluated similar to the agenda in NEXPERT and CLIPS (an expert system developed by NASA Johnson Space Center). The inference strategy for SCL is now user selectable for either

depth first evaluation or evaluation using an agenda.

The SCL inference engine time-shares with the SCL command interpreter, i.e., rules are evaluated in parallel with script execution. This method allows scripts to execute at timed intervals and set database variables that control the mode of the attitude determination tasks. Changes to these variables result in corresponding rules to be executed, allowing timing synchronization between the scripts and rules.

The differing inference strategies between the two systems provided some interesting results. Statistics for rule execution for the two systems were collected. The NEXPERT rule base required fewer rules to be executed because of the prioritized agenda. Although the SCL expert system executed some rules more than once, the end result (reflected in telemetry variables) was identical for both systems. For a given test scenario, the SCL system was able to evaluate more rules per second.

Object Representation

The knowledge base is centered around a related group of objects. Both NEXPERT and SCL require the knowledge engineer to describe the ACE data points for both commands and telemetry in terms of objects. Rules in the knowledge base respond to changes in the values of these objects and may induce changes in one or more other objects.

Both SCL and NEXPERT are generic expert systems, that is, neither have any ties to a given spacecraft. SCL describes telemetry sensors, command actuators, and derived items in a database. Telemetry sensors are physical sensor readings, while derived items are data points that are calculated based on readings from one or more sensors. Command actuators are data points that are used to activate/deactivate relays, or serial data words which are interpreted by local or remote command functions. SCL groups these items by a sorting field called the subsystem.

NEXPERT allows data points to be defined in terms of classes, subclasses, objects, subobjects, and properties.

The SCL database was designed to support a real-time system and was purposely designed to have a limited amount of data abstraction in terms of object oriented programming. The tradeoff was felt to be necessary due to memory and real-time considerations when running in an embedded environment.

Testing the Knowledge Base

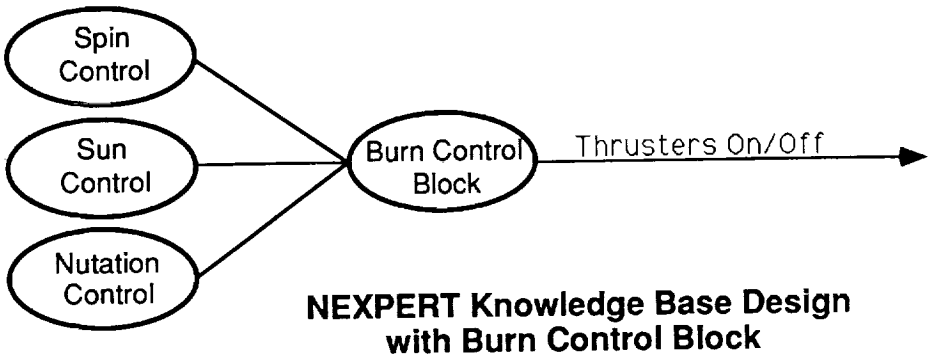
Once the knowledge base was developed, the rules and scripts must be tested and debugged. The NEXPERT system provides a debugger similar to those used in conventional programming systems. Break points may be set on rules and data objects contained in the knowledge base. When a break point is encountered, data objects may be examined and/or modified.

SCL allows the user to examine and modify the knowledge base data points from its command window. Much of the SCL syntax that is used in scripts and rules is also valid from the command window. To implement a break point, a "stop inference engine" statement needed to be placed in the script or rule that a break point is desired. A full featured source level debugger is currently under development for a future release of SCL.

SCL also allows the user to trace all script and rule execution or individual scripts and rules may be selected for tracing. Several levels of tracing are supported, as well as tracing on all or selected database points.

Expert System Verification

The resulting expert systems were tested for response and accuracy and compared to the actual flight software. To verify the results, the ACE flight software dynamic model was used in the same manner as it was to acceptance tests the flight software. The following tests were used to



NEXPERT Knowledge Base Design with Burn Control Block

verify the closed-loop control algorithms for spin period, sun angle, and active nutation:

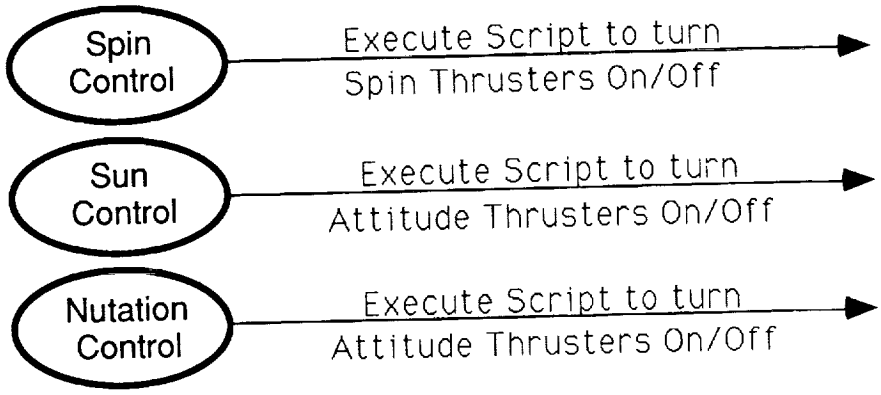
- Sun Angle Correction - begin at 75° angle, maintain to 90° +/- 10°.
- Sun Angle Correction - begin at 105° angle, maintain to 90° +/- 10°.
- Spin Period Correction - begin at 3.5 r.p.m., maintain 5.0 r.p.m. +/- 10%.
- Spin Period Correction - begin at 5.5 r.p.m., maintain 5.0 r.p.m. +/- 10%.
- Nutation Correction - begin at 0.1° cone angle, maintain 0.25° cone angle.
- Nutation Correction - begin at 10.0° cone angle, maintain 0.25° cone angle.
- Nutation and Sun Angle Correction.
- Spin Period and Sun Angle Correction.
- Spin Period, Nutation, and Sun Angle Correction.

The closed loop control of the spin period and sun angle worked equally well using both expert systems. Both returned the same results as those reported by the ACE flight software. While testing the nutation control with NEXPERT, problems were detected while attempting to initiate a thruster maneuver upon encountering a zero-crossing in the nutation filter. The problem was traced to the design of the NEXPERT knowledge base and the amount of time required to process the rules upon encountering a zero crossing. Because of this, the NEXPERT knowledge base was not able to correct the nutation.

To correct this problem, the NEXPERT knowledge base was modified to reduce the number of rules that needed to be evaluated at the zero crossing. This allowed the active nutation to be corrected to the desired level. The NEXPERT knowledge base corrected the low nutation as efficiently as the ACE flight software, however, an additional 400 millisecond burn was required of the 30 lb. thruster pair to correct the 10° cone angle.

Having seen the difficulties in achieving accurate timing for thruster burn start and stop, the knowledge base architecture was redesigned to take advantage of the SCL scripting capability. Previously using NEXPERT, the spin control algorithm, the sun control algorithm and the nutation control algorithm fed the burn control block. The burn control block controlled the firing of all thrusters. The SCL knowledge base was redesigned to allow rules that control the spin, sun, and nutation control and to execute scripts directly which initiate and terminate the appropriate thrusters.

This design allowed the SCL knowledge base to correct all nutation as effectively as the actual ACE flight software. The NEXPERT knowledge based could also have been redesigned, but it was not deemed to be sufficiently beneficial.



SCL Knowledge Base Design with Scripts

Conclusions

The ACE expert systems proved that expert system technology can be applied to classic control loop algorithms. The speed, accuracy, and memory requirements can be met using a more modern approach. However, a commercial product was not deemed feasible. The extensive memory usage and real-time limitations of the commercial product made it a poor choice for this application. NEXPERT's inability to keep pace with the control loops made it an unacceptable alternative.

SCL proved to be a capable of performing all ACE monitor and control functions at least as well as the actual flight code. SCL is a specialized product designed for satellite command and control systems. Since SCL was designed to run in real-time embedded environments, overall memory usage has been minimized, and SCL does not rely on dynamic memory allocation (since it will eventually cause fragmentation.) SCL was designed to run on spacecraft with radiation-hardened processors in the 1.5-3 MIPS range and tight memory requirements. With the skyrocketing cost of rad-hard error correcting memory, the SCL designers chose not to include more extensive object-oriented features.

Because of the favorable results of the ACE effort, the SCL expert system was chosen as the embedded flight controller software for a future NRL satellite to be used on a national program. SCL will be used for the satellite as well as the dispenser. SCL has also been chosen as the ground based expert system at NRL ground stations to support upcoming missions as well as provide advisory systems for existing satellite systems.

The expert system approach allows much greater productivity for the engineers since they are working in a very high level language and have sophisticated development tools available. The systems can also be modeled and tested on desktop workstations rather than having to use specialized test fixtures.

Because SCL is not a mass-marketed commercial expert system, changes could be made to the SCL system based on lessons learned during the ACE project:

- The agenda inference strategy was added.
- A source-level debugger is under development.
- The SCL development Man Machine Interface is being ported to Motif/X-Windows to run on many popular workstations and X-terminals.

- A graphics interface for SCL is being developed to allow the developer to view relationships between database objects, rules, and scripts.
- Several extensions to the SCL syntax were requested and have been added to the system.

Applied Expert System Technology

In the past, expert system technology has been difficult to use, expensive, and ran only on specialized hardware. Modern expert systems are available on a wide variety of processors and have become an efficient and cost-effective solution for systems development. The expert system technology can easily be merged with conventional technology to allow simplification of system development. By developing realistic dynamic simulations, rapid prototyping and modeling of subsystems and entire systems is practical. Developers can checkout complex systems in a desktop environment and can find potential problems early in the development cycle. These simulations can be part of a test bed that will not only support control system software development and validation but the same system can be used for training. This expert system technology can be merged with other types of technology such as databases, spreadsheets, graphics, hypermedia, animation, sound, and conventional code.

Expert systems can also aid in the design and development of systems by providing a reasoning mechanism that models human reasoning, providing data representation that more closely models the "real world", and allowing generic systems to be developed and be reused, while localizing the application specifics in the knowledge base.

SCL

SCL is unique in the area of command and control because of its small size, ease of use, versatility, reusability, and adaptability. SCL offers, in one package, the following capabilities:

- Real-time command and control - not a static query expert system (Medical, Financial, etc.).
- Complete integrated environment with re-usable software allows SCL to be used for diverse applications.
- Supports embedded and distributed applications.
- Scripting capability for real-time command and control, monitoring, modeling, simulations, and training.
- Easy to learn, excellent Man/Machine Interface (MMI).

Compact sizing as well as the abstraction of hardware specifics allows the standard SCL kernel to be used on a variety of hosts for embedded processors as well as mini and micro-computer and workstation applications.

SCL can be used for software simulations of subsystems as well as an entire spacecraft, systems modeling, and training.

DOD, NASA, and the commercial sector will be able to use SCL for any applications requiring smart subsystem control and monitoring. These applications could include uses in support of the Space Station, robotics, future Lunar and Mars missions, as well as commercial applications requiring real-time smart control system process monitoring (petrochemicals, manufacturing, utilities, etc.).

Acknowledgments

We would like to thank the following people for their contributions to this paper: Dave Schriftman, Jim Van Gaasbeck, Patrice Cappelaere, and Dean Oswald.

References

1. Buckley, Brian and Wheatcraft, Louis: "Spacecraft Command Language - A Smart Control System", Interface and Control Systems, Melbourne, Fl., Barrios Technology, Houston, TX., March 1991

2. Van Gaasbeck, James: "Technical Overview of the Spacecraft Command Language" Naval Research Laboratory, Washington D.C., 1991

3. Interface and Control Systems: "SCL User's Guide", Naval Research Laboratory, Washington D.C., 1990

4. Buckley, Brian and Wheatcraft, Louis: "Rapid Prototyping of a Spacecraft Controller", JAIPCC Symposium, Houston Tx., March 1991

5. Buckley, Brian and Wheatcraft, Louis: "Spacecraft Simulations with a re-usable Smart Control System", JAIPCC Symposium, Houston Tx., March 1991