The Implementation of Satellite Attitude Control System Software Using Object Oriented Design

W. Mark Reid
NASA Goddard Space Flight Center, Greenbelt, MD 20771
William.M.Reid.1@gsfc.nasa.gov 301-286-6088

William Hansell, Tom Phillips
the Hammers Company, Greenbelt, MD 20771

Abstract. NASA established the Small Explorer (SMEX) program in 1988 to provide frequent opportunities for highly focused and relatively inexpensive space science missions. The SMEX program has produced five satellites, three of which have been successfully launched. The remaining two spacecraft are scheduled for launch within the coming year. NASA has recently developed a prototype for the next generation Small Explorer spacecraft (SMEX-Lite). This paper describes the object-oriented design (OOD) of the SMEX-Lite Attitude Control System (ACS) software. The SMEX-Lite ACS is three-axis controlled and is capable of performing sub-arc-minute pointing. This paper first describes high level requirements governing the SMEX-Lite ACS software architecture. Next, the context in which the software resides is explained. The paper describes the principles of encapsulation, inheritance, and polymorphism with respect to the implementation of an ACS software system. This paper will also discuss the design of several ACS software components. Specifically, object-oriented designs are presented for sensor data processing, attitude determination, attitude control, and failure detection. Finally, this paper will address the establishment of the ACS Foundation Class (AFC) Library. The AFC is a large software repository, requiring a minimal amount of code modifications to produce ACS software for future projects.

Acronyms

ACS	Attitude Control System
AC	ACS Control (software task)
AFC	ACS Foundation Class
AIAA	American Institute of Aeronautics and Astronautics
AM	ACS Models (software task)
BC	MIL-STD-1553B Bus Controller
BMC	Bulk Memory Card
C&DH	Command and Data Handling
CI	Command Ingest (software task)
COMM	Communications
CSS	Coarse Sun Sensor
DS	Data Storage (software task)
DSS	Digital Sun Sensor
EDAC	Error Detection and Correction
FAST	Fast Auroral Snapshot Explorer
FDH	Failure Detection and Handling
FOV	Field of View
GPS	Global Positioning System
HK	Housekeeping (software task)
IC	Instrument Control (software task)
I/F	Interface
IRWA	Integrated Reaction Wheel Assembly
LC	Limit Checker (software task)
MIL-STD	Military Standard
MTB	Magnetic Torquer Bar
NASA	National Aeronautics and Space Administration
OOD	Object-Oriented Design
PCI	Peripheral Component Interconnect
PROM	Programmable Read Only Memory
RT	MIL-STD-1553B Remote Terminal
SAMPEX	Solar Anomalous and Magnetospheric Particle Explorer
SC	Stored Command (software task)

S/C	Spacecraft
SFP	SMEX-Lite Flight Processor
SH	Scheduler (software task)
SMEX	Small Explorer
SWAS	Submillimeter Wave Astronomy Satellite
TAM	Three-Axis Magnetometer
TO	Telemetry Output (software task)
TRACE	Transition Region and Coronal Explorer
TRMM	Tropical Rainfall Measurement Mission
WAES	Wide Angle Earth Sensor
WIRE	Wide-Field Infrared Explorer
RXTE	Rossi X-Ray Timing Explorer
ZSP	Zenith Sun Point

Introduction

NASA's Small Explorer (SMEX) program builds small, relatively inexpensive spacecraft. These spacecraft provide opportunities for performing highly focused space science. The typical SMEX spacecraft has a mass of approximately 180 to 250 kg. The typical development schedule for each SMEX spacecraft has been three years from start to launch. The SMEX program has produced five satellites: Solar Anomalous and Magnetospheric Particle Explorer (SAMPEX), Fast Auroral Snapshot Explorer (FAST), Submillimeter Wave Astronomy Satellite (SWAS), Transition Region and Coronal Explorer (TRACE) and Wide-Field Infrared Explorer (WIRE). SAMPEX, FAST and TRACE are on-orbit. WIRE is scheduled for launch in September of 1998, and SWAS is scheduled for launch in January of 1999. In each of these missions, the Attitude Control System (ACS) software was written using a modular procedural design.

The SMEX program is now looking to the future with the development of the next generation SMEX spacecraft bus known as SMEX-Lite. Current program goals reduce spacecraft development time as well as development costs, while maintaining mission reliability and enhancing performance. To accomplish these goals, there is a heavy reliance on flight software and software reuse. To obtain increased reuse of the flight software, an object-oriented design approach was taken.

This paper is principally concerned with the design and development of the ACS flight software. This design parallels the component oriented "plug-and-play" architecture of the spacecraft, allowing increased flexibility for future missions. This paper will provide details of the SMEX-Lite ACS software design, demonstrating its functionality and flexibility. Additionally the paper will show how the establishment of an ACS Foundation Class library will provide increased reuse, for SMEX-Lite as well as other NASA programs.

SMEX-Lite Architecture

The SMEX-Lite spacecraft was designed and developed to have a "plug-and-play" architecture. This allows components to be added or removed as required for a specific science mission, with little redesign. The entire spacecraft has been designed to have independent components or subsystems, as indicated in Figure 1 below.

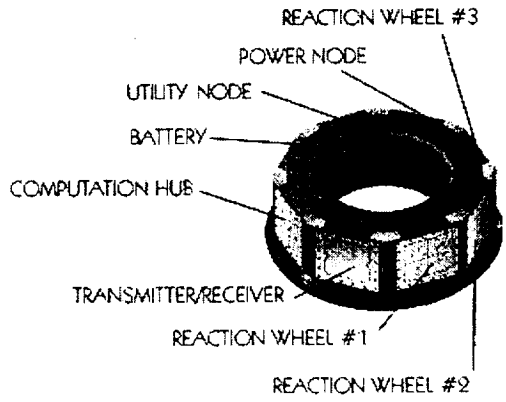

Figure 1: SMEX-Lite Spacecraft

The central spacecraft computer for SMEX-Lite is the Computation Hub. Various spacecraft components interface with the Computation Hub via a MIL-STD-1553B data bus. Figure 2 provides a high level design of the SMEX-Lite ACS system. This figure shows the components that are supported in the current SMEX-Lite design and indicates how the various ACS components communicate over the bus. Additional components can be added to support specific future mission requirements.

Integrated Reaction Wheel Assemblies. The current SMEX-Lite ACS system supports three Integrated Reaction Wheel Assemblies (IRWAs). Each IRWA is a completely self-contained assembly, containing flywheel, power electronics, tachometer, and MIL-STD-1553B interface. Each IRWA acts as both a sensor and an actuator. The ACS software uses tachometer information provided by each IRWA to determine the wheel's speed or momentum. This information can then be used to derive changes in spacecraft body rates. Similarly, if a change in spacecraft attitude is required, commands are sent to the IRWAs to modify their spin rates, causing the spacecraft body to react. If a fourth IRWA is required, it can simply be added to the MIL-STD-1553B data bus with little changes to the existing spacecraft hardware. Only the ACS software within the Computation Hub would need to be modified.

Utility Node. The SMEX-Lite ACS system also supports multiple sensors through an electronics box known as the Utility Node. The Utility Node provides the interface between these sensors and the MIL-STD-1553B. The SMEX-Lite implementation of the Utility Node provides interfaces for six Coarse Sun Sensors (CSSs), one Digital Sun Sensor (DSS), a Three-Axis Magnetometer (TAM), and an Earth Sensor. Additionally, the Utility Node supports actuators. For SMEX-Lite, it

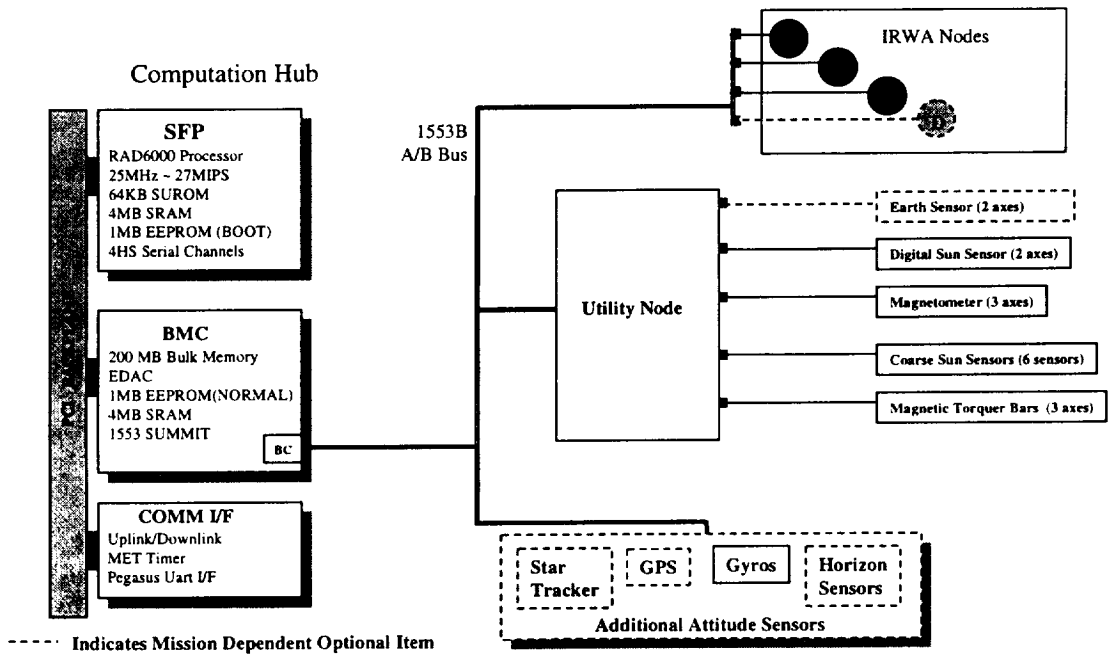

Figure 2: SMEX-Lite ACS System Design

provides an interface for three Magnetic Torquer Bars (MTBs). The Utility Node also supports expansion if additional sensors or actuators are required.

Coarse Sun Sensors. The six CSSs are typically mounted in pairs on each of the spacecraft's primary axes to provide full sky coverage. These sensors, which are supplied by Adcole, each have a +-85 degree field of view (FOV).[1] The CSSs are analog devices providing eclipse detection and a measure of the sun's intensity. The ACS software uses this coarse sun information for initial sun acquisition control and for attitude determination in science modes.

Digital Sun Sensor. The DSS, which is also supplied by Adcole, has a +- 64 degree FOV and a resolution of 0.5 degree.[2] It is typically mounted on the sun pointing side of the spacecraft and provides a more precisely measured sun vector for sun acquisition control and attitude determination in science modes. The ACS software automatically switches to using the DSS once the Sun is in the DSS FOV.

Three Axis Magnetometer. The TAM provides a measure of the earth's magnetic field vector at the spacecraft. The magnetic field vector is used by all control modes. An additional ACS control mode is used to perform an on-orbit calibration of the magnetometer. This control mode sends commands to the MTBs and measures the effects on the magnetic field readings. It then computes a magnetic compensation matrix, which is used to remove these effects.

Earth Sensor. The Earth Sensor is supported only by the Utility Node hardware. The Utility Node provides an interface for three signals, corresponding to the outputs of a Wide Angle Earth Sensor (WAES). The WAES, which is supplied by the Servo Coporation, was developed for the WIRE mission. The baseline SMEX-Lite ACS software has no requirements for using the WAES. It was decided, therefore, not to implement the WAES in the software. However, the object-oriented design of the ACS software greatly simplifies the addition of the WAES for future projects.

Magnetic Torquer Bars. MTBs are mounted along each spacecraft body axis. The MTBs provide rate damping and sun pointing control during initial sun acquisition. They are also used for momentum unloading in science pointing modes. The MTBs supported during the SMEX-Lite prototype testing have a range of +- 60 A-m^2, corresponding to the Ithaco Torque Rods used in earlier SMEX missions.

Other Attitude Sensors. Additional sensors may be added to the SMEX-Lite system as required. The baseline SMEX-Lite software provides support for a gyro package containing three two-axis gyros. These are mounted parallel to each spacecraft body axis, and provide a completely redundant configuration. The gyro data is used to measure spacecraft body rates. This information is used for spacecraft attitude determination in science pointing modes. The gyros used in prior SMEX missions are Bell Textron tuned restraint inertial gyros. It is anticipated that these gyros will have a drift of 0.6 deg/hr and noise of less than 4.5 arcsec/sec.[3] If additional sensors such as a Star Tracker, GPS receiver, or Horizon sensor are required, the system is designed to allow these sensors to be inserted with minimal rework of the existing design.

ACS Software Context

The SMEX-Lite Flight Processor (SFP), which is located within the spacecraft's Computation Hub, supports multitasking software running under VxWorks. It operates in one of two modes, Boot Mode and Normal Mode. Upon power up, the SFP begins executing in Boot

Mode. In this mode, the processor loads tasks from Programmable Read Only Memory (PROM). These tasks provide the basic command, telemetry, software management, and health and safety monitoring required to maintain spacecraft safety. With the absence of any secondary processor, it is required that the SFP have an ACS task in Boot Mode that performs initial sun acquisition and safehold functions. This task uses data from the Utility Node and the IRWAs to command the IRWAs and MTBS, pointing the solar arrays toward the sun and damping the spacecraft body rates. The Boot Mode task also performs some failure detection and handling (FDH). This FDH includes checking the spacecraft sun angle, spacecraft body rates and wheel power status. The Boot Mode ACS allows only minimal commanding and generates telemetry necessary for spacecraft health and safety checks. The initial Boot Mode software may not be modified on orbit.

After initial acquisition, the spacecraft is commanded into Normal Mode. Figure 3 details the Normal Mode ACS software context. The Normal Mode ACS software consists of two tasks, the ACS Control (AC) task and the ACS Models (AM) task. Additional ACS software resides in the system which is used by both AC and AM. This includes math routines and other shared utility functions.

The AM Task. The ACS Models task is responsible for propagating the orbit ephemeris models. The AM task produces inertial vectors, which are sent to the AC task at a 1Hz rate. These include a Solar Inertial Vector, Magnetic Field Inertial Vector, Spacecraft Position Vector, and Spacecraft Velocity Vector.

The AC Task. The main software task for the ACS is the AC task. It receives sensor data from the hardware, processes the data, and generates the actuator commands to perform active spacecraft control. AC has important

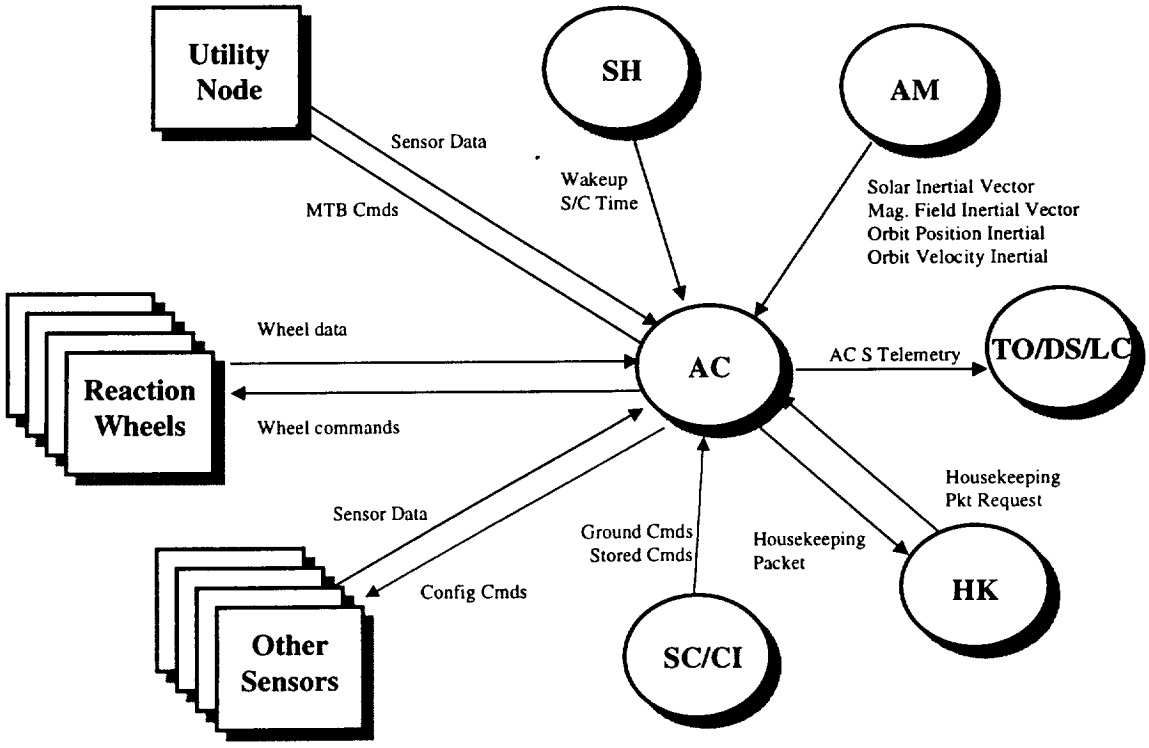

Figure 3: SMEX-Lite Normal Mode ACS Software Context

interfaces with several additional Command and Data Handling (C&DH) tasks. The Scheduler task (SH) provides a wake-up packet to AC at the desired execution frequency. The baseline execution frequency for SMEX-Lite is 10 Hz. The Stored Command (SC) and Command Ingest (CI) tasks route stored sequences of commands or ground commands to AC. The Housekeeping (HK) task monitors AC's health, while the Telemetry Output (TO) and Data Storage (DS) tasks route telemetry from the AC task to the ground and bulk memory.[4] The Limit Checker (LC) task provides additional monitoring of the AC task's telemetry. This task is used to place the spacecraft into a safe configuration if serious limit violations occur.

SMEX-Lite ACS Software Design

Written in C++, the SMEX-Lite ACS software takes advantage of three important concepts of object-oriented programming. They are encapsulation, inheritance, and polymorphism. These OOD principles provide a framework for common software interfaces and increase software integrity and reuse. Four areas of the SMEX-Lite ACS software, which make use of these design principles, are Sensor Data Processing, Attitude Determination, Attitude Control, and Failure Detection and Handling (FDH).

Encapsulation. A C++ class encapsulates information by bundling an object's attributes (data) with its methods (functions) and treating them as a single entity. This principle is also often referred to as "information hiding" and protects the integrity of the data.[5] Each SMEX-Lite component has been designed as an independent, encapsulated object. The benefit of this encapsulation is that the object will maintain its own data, perform its own calculations, and return its product through a standard interface. Other objects will only have access to member data through specific methods, making the external interface standard and streamlined. Another benefit of this encapsulation is that each of the components accepts all of their required data in the form of parameters, making them independent of the system as a whole for their information. This is a break with past SMEX implementations of space flight software, where data was maintained in global structures and each component had to be totally aware of the structure and naming.

Inheritance. Object-oriented programming also provides a mechanism for abstracting data that is common to various classes or objects. This concept, known as class inheritance, allows for derived classes to use data and operations that have been previously defined in a base class.[6] The basic design of the SMEX Lite software was to ensure that all common elements of a hierarchy were well abstracted, leaving only the most specific and unique attributes to the concrete instantiation. Any new implementations of components will require coding only the most specific details relating to the hardware or algorithm.

Polymorphism. The third Object-oriented design principle that influenced the SMEX-Lite ACS software is polymorphism. This principle allows objects from a variety of classes to respond to the same message. The message's receiver is determined dynamically at runtime.[7] The SMEX-Lite software, although not truly polymorphic, was designed with this principle in mind. Through overloading methods, the software allows for a common interface to all objects with similar attributes. The SMEX-Lite software provides methods with the same name within multiple classes or objects. These methods perform different operations based on which object is being referenced. For example both the CSS and DSS objects provide a method for computing the sun vector from the raw data. The actual calculations required for each sensor, however, are very different.

Sensor Data Processing. The area of the SMEX-Lite software that most clearly takes advantage of the principle of inheritance is sensor data processing. The sensor data processing software performs the function of converting raw data from the sensor hardware into engineering data used for spacecraft attitude determination and control. This software is used in both the spacecraft Boot Mode and Normal Mode.

Figure 4 shows how the many sensor objects are related through an inheritance hierarchy to an abstract base class cSensor. The cSensor base class provides all of the attributes and methods that are common to all sensors. For SMEX-Lite all sensors are given the attributes of a power state, online state and a flag indicating data validity. Any inherited sensor class will have these attributes as well as methods for retrieving or setting these data items.

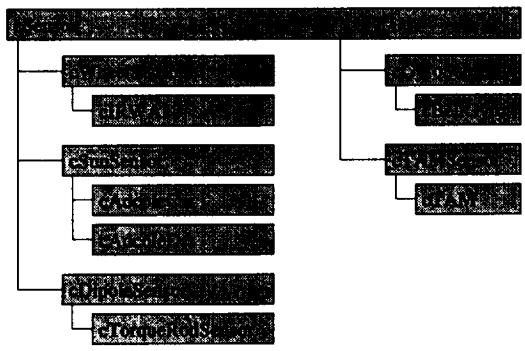

Figure 4: Sensor Classes

The next level of abstraction shows that there are various types of sensors. For example all sun sensors which are derived from the cSunSensor class will have the attributes of an illumination flag, alignment matrix, sun vector (in S/C body frame) and a sun angle. Additionally any derived class will have methods for setting, referencing, or computing these data items.

At the most concrete level, specific details of an individual sensor are defined. For any CSS object which is instantiated using the cAdcoleCSS class, these specific attributes include scale factors, biases, voltage thresholds (required for illumination) and a maximum voltage level. The CSS object would also contain the specific method for computing engineering data from the raw data input. Similarly, any DSS object would have the attributes and methods specific to a DSS. These include X-axis and Z-axis scale factors and biases, DSS test threshold, and head identifier. It would also contain the specific routine for computing engineering data.

This same level of abstraction was placed into each of the derived sensor classes, even though most abstract sensor classes are used by only one specific sensor type. For example, there is an abstract class for wheel sensors even though SMEX-Lite has only one type of reaction wheel, the IRWA. This abstract wheel class acts as a placeholder for wheels that may be added on future spacecraft. It was assumed that all wheels would have some common attributes such as a wheel speed, and would also have a method for retrieving this data. Placing this framework into the SMEX-Lite software increases the reusability of this software for other NASA projects.

Attitude Determination. SMEX-Lite performs attitude determination as part of the Normal Mode ACS task. The Attitude Determination subsystem was designed around a core 'processor' object that communicates with 'attitude determination' objects. The processor determines when an attitude determination object will be called and what variables it will receive. All interfaces to the rest of the system are sent through the processor object. The data that an object requires to perform computations is either stored locally within the object or sent to it from the processor as a parameter. There is no

global data to be accessed. This greatly enhances system reliability since various methods can be totally isolated and tested independently.

All of the attitude determination objects operate with the same interface; they receive their data either in a constructor or as an argument, and their data is returned through a common function name. For example, all objects execute using a run() method and output data with a ref() method. This consistent object interface makes it very easy to reuse the software in different projects.

All objects that have the same function are derived from the same base class, as shown in figure 5. This means that the data elements that each of the objects inherits are the same for each of the concrete classes. This greatly simplifies code development since it is known what will be returned and what the variables are called. For example, a specific filter class will inherit a data valid flag and a vector data type. Also included in the inheritance are member functions or methods by which these variables are extracted. When the programmer creates a new filter class from this class much of the software is already implemented. Only the specific processing for the new class must be coded.

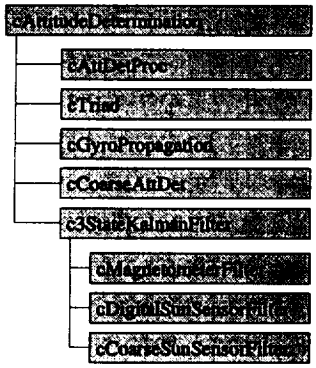

Figure 5: Attitude Determination Classes

The Attitude Determination software was designed as a set of autonomous building blocks, each being able to function as an independent unit. Each of the attitude determination objects could be placed into a different application with little coding effort and the proper results would be returned.

cTriad is derived from cAttitudeDetermination, which maintains the attitude solution and methods for extracting either a Quaternion or a Direction Cosine Matrix. The cTriad class will take in two pairs of vectors (inertial frame, body frame) and compute the estimated attitude.

The cGyroPropagation class is derived from cAttitudeDetermination and cCovariance. The Gyro Propagation class will take in the delta angle from the Gyro Processor along with the current quaternion and the current error (covariance) and compute a new attitude and error. Again the methods for extracting the attitude and error (covariance) are contained in the base classes.

cMagnetometerFilter, cDigitalSunSensorFilter, and cCoarseSunSensorFilter all take in the same arguments, the inertial vector from the Models task, the measured vector, the current attitude, and the current error. The error is updated by the Kalman Filter and applied to the attitude to get a new solution.

Attitude Control. The SMEX-Lite ACS software contains four different classes of controller objects for controlling spacecraft attitude. These objects contain the control laws that compute torque and dipole commands, which are sent to the IRWA and MTB actuators. These controllers are executed based on the current spacecraft mode and ACS control mode. These classes are shown in figure 6.

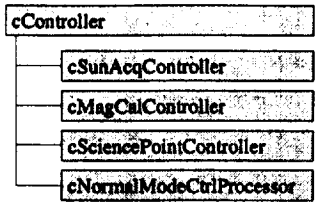

Figure 6: Attitude Control Classes

In spacecraft Boot Mode, the ACS task contains a single controller. This controller is used for sun acquisition and rate damping, and is referred to as the Sun Acquisition or Safehold controller. It is implemented in the class 'cSunAcqController'. This controller uses all of the reaction wheels to point the spacecraft solar arrays toward the sun. It also uses the magnetic torquer bars to establish and maintain the system momentum bias while reducing the spacecraft body rates. This Boot Mode sun acquisition acts as the initial acquisition controller immediately after launch as well as a safety net in the event that the Normal Mode software fails to perform as expected.

Once the spacecraft is in Normal Mode, the ACS provides the flexibility of additional control modes. The three Normal Mode ACS control modes are Sun Acquisition, Magnetic Calibration, and Science Point. Each of these modes has a corresponding controller, cSunAcqController, cMagCalController, and cSciencePointController. Yet another controller object decides which of these controllers will be used, based on the desired ACS control mode. This is referred to as the Normal Mode Control Processor, and is implemented in cNormalModeCtrlProcessor.

Magnetic Calibration mode is used to fine-tune the onboard magnetic compensation matrix. This matrix is used by the TAM object to correct TAM readings, which are corrupted by the magnetic field produced by the MTBs. During this calibration, the reaction wheels are controlled to constant speeds, and a sequence of dipole commands is sent to the MTBs. A sequence of magnetometer readings is taken and the updated magnetic compensation matrix is computed. The duration of the calibration is approximately six seconds.

The Science Point Control Mode is used to perform mission specific science targeting. The baseline for SMEX-Lite was taken from WIRE's Zenith Sun Point (ZSP) mode. The controller uses utility node, IRWA, and gyro data to point the spacecraft +y-axis toward the sun and the spacecraft +z-axis toward zenith. During this mode, the ACS software also performs attitude determination, as needed by the ZSP controller. If a more robust Science Point Controller is needed, only this one object would need to be modified.

Failure Detection and Handling. The FDH subsystem is responsible for executing ACS performance and limit failure checks to monitor the health and safety of the spacecraft. The FDH subsystem was designed around the FDH Check object. Each FDH Check object contains the check limits, the actions which will be taken if excessive failures occur, any counters needed to keep track of successive failures, and the logic to perform the check itself.

Each FDH check object will, when called upon, perform the check that determines if a certain value is within its derived bounds. Upon a failure, a check object will increment a counter. If no failure is found, the counter is reset to zero. If the counter exceeds the objects' Maximum permitted error count, then the object takes corrective action.

FDH failure actions are driven by mission requirements. Upon a failure condition, some FDH Checks may do nothing more than post an event status message highlighting the issue for

review on the ground. Other FDH check objects will request a change of an ACS mode to a lower, safer mode. Some checks execute a MIL-STD-1553B bus command to power on a reaction wheel. These actions are different for each FDH check.

The FDH subsystem exists for both the Normal and Boot mode. Each of the required FDH checks is controlled through a centralized FDH processor object. The Normal mode FDH processor performs each check once per cycle (10 Hz). Boot mode FDH is performed once per boot mode cycle (2 Hz). Different checks are performed for each of the modes. Each check may have different constraints, different tolerances, and may take different corrective actions. Figure 7 shows the FDH checks which are performed for SMEX-Lite.

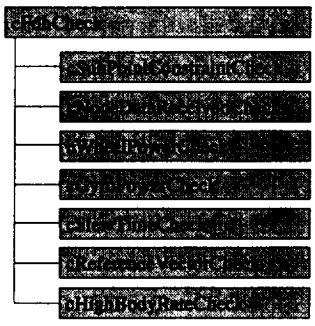

Figure 7: FDH Check Classes

FDH checks, which are performed during Boot Mode, include the following:

- Sun Pointing Constraint Check
- Utility Node Data Received Check
- Wheel Node Data Received Check
- Wheel Node Power Check

The Sun pointing constraint check ensures that the spacecraft is oriented within limits relative to the sun during sun acquisition mode. The Utility Node data received check is made to ensure data is still being received from the utility node. Similarly, the Wheel Node data received check ensures that data is still being received from each reaction wheel. The Wheel Node power check is made to ensure that the each wheel is still powered on. If a wheel is powered off for an extended number of cycles, the FDH processor will issue a MIL-STD-1553B bus command to power on the wheel.

In addition to the checks performed in Boot Mode the Normal Mode FDH performs the following checks:

- Gyro Power Check
- Excess Body Rate Check
- Gyro Node Data Received Check
- Inertial Reference Vectors Check
- Slew Execution Time Check
- Sun Pointing during ZSP Check

The gyro power check ensures that the correct number of gyros is powered on for Normal Mode operations. The excess body rate check is made to ensure that the spacecraft body rate does not exceed constraints. The Gyro Node data received check verifies that data is still being received from the gyro node. The inertial reference vectors check determines if all of the model validation flags are set TRUE, indicating that the reference vectors received from the AM task are valid. The slew execution time check ensures that a spacecraft slew command is completed within the allowed time constraint. The Sun pointing during ZSP check performs the additional check to ensure that the spacecraft is oriented within limits relative to the sun during Zenith Sun Pointing mode.

ACS Foundation Class Library

The ACS software developed for the SMEX-Lite prototype has been used to establish an ACS Foundation Class Library and is now serving as a baseline for future spacecraft. The ACS Foundation Class Library consists of objects designed for building Attitude Control Software Systems. Many of the objects found within this library are generic and can be

reused without code modification. Some objects, however, are mission unique and would require some modification to meet mission specific requirements.

The goal is for future projects to reuse many of the abstract classes and much of the existing ACS software framework. New mission specific classes can be derived from existing classes to provide reuse while allowing for increased flexibility. The ACS Foundation Class Library should be viewed as an ever-changing set of classes, which grows as future missions are developed. Future development efforts must wisely choose which areas of the software should be reused and which areas require modifications or even new classes to be developed.

Summary

The SMEX-Lite ACS software takes advantage of the Object-oriented design principles of encapsulation, inheritance, and polymorphism to provide a robust and highly reusable software library. This paper has detailed the SMEX-Lite software design and functionality. This software design provides a framework which will increase software integrity and reduce development time and cost. Future spacecraft can draw on the resources developed for SMEX-Lite to create a highly focused yet flexible and reusable ACS system.

Acknowledgments

The authors of this paper wish to extend a special thanks to Mark Anderson for getting this paper started and for all of his leadership during the SMEX-Lite development effort. Thanks also go to those who have helped by reviewing this paper, including Ken Barnes, Tom Correll, Susanne Strege, and Miriam Wennersten.

References

1. Fennell, M., V. Untalan, Dr. M. Lee, "The Attitude Control System Design for the Wide-Field Infrared Explorer Mission," Proceedings of the 11th Annual AIAA/USU Conference on Small Satellites, September 1997.

2. Ibid.

3. Ibid.

4. Barnes, K., C. Melhorn, T. Phillips, "The Software Design for the Wide-Field Infrared Explorer Attitude Control System," To be published in the proceedings of the 12th Annual AIAA/USU Conference on Small Satellites, September 1998.

5. Ford, W., W. Topp, Data Structures with C++, Prentice-Hall Inc., 1996, p. 7.

6. Ibid., pp. 11-13.

7. Ibid., pp. 34-36.

Authors' Biographies

Mark Reid is currently the ACS Software Lead Engineer for the SMEX-Lite spacecraft. He has a B.A. in mathematics from Western Kentucky University with a minor in physics. He is pursuing a M.S. in computer science from Johns Hopkins University. In addition to his work on SMEX-Lite, he has developed ACS flight software for the Spartan program, and the SAMPEX, RXTE, TRACE and WIRE spacecraft.

William Hansell graduated from The University of Maryland, College Park in 1989. His Aerospace software experiences include developing ground support

software and attitude control software. In addition to developing ACS software for NASA's SMEX-LITE project, Mr. Hansell has worked on the MAP, WIRE, and Spartan programs. He has also written data management software for TRMM. Prior to coming to NASA's Goddard Space Flight Center in 1994, Mr. Hansell worked with the US Navy, developing software for the Naval Sea Systems Command.

Tom Phillips graduated from Davis & Elkins College with a BSCS in 1986. His Aerospace software experiences include developing ground support software and attitude control software. In addition to developing ACS software for NASA's SMEX-LITE project, Mr. Phillips has worked on NASA's TRMM, RXTE, WIRE, and Spartan programs. He has also written hardware verification software for the Air Force STEP program. Prior to coming to NASA's Goddard Space Flight Center in 1992, Mr. Phillips worked with the US Navy, developing shore based communication software

THE IMPLEMENTATION OF SATELLITE ATTITUDE CONTROL SYSTEM SOFTWARE USING OBJECT ORIENTED DESIGN

Mark O. Anderson, Mark Reid
NASA Goddard Space Flight Center
Greenbelt, Maryland

Derek Drury, William Hansell, Tom Phillips
the Hammers Company
Greenbelt, Maryland

Abstract

NASA established the Small Explorer (SMEX) program in 1988 to provide frequent opportunities for highly focused and relatively inexpensive space science missions that can be launched into low earth orbit by small expendable vehicles. The development schedule for each SMEX spacecraft was three years from start to launch. The SMEX program has produced five satellites; Solar Anomalous and Magnetospheric Particle Explorer (SAMPEX), Fast Auroral Snapshot Explorer (FAST), Submillimeter Wave Astronomy Satellite (SWAS), Transition Region and Coronal Explorer (TRACE) and Wide-Field Infrared Explorer (WIRE). SAMPEX and FAST are on-orbit, TRACE is scheduled to be launched in April of 1998, WIRE is scheduled to be launched in September of 1998, and SWAS is scheduled to be launched in January of 1999. In each of these missions, the Attitude Control System (ACS) software was written using a modular procedural design. Current program goals require complete spacecraft development within 18 months. This requirement has increased pressure to write reusable flight software. Object-Oriented Design (OOD) offers the constructs for developing an application that only needs modification for mission unique requirements.

This paper describes the OOD that was used to develop the SMEX-Lite ACS software. The SMEX-Lite ACS is three-axis controlled, momentum stabilized, and is capable of performing sub-arc-minute pointing. The paper first describes the high level requirements which governed the architecture of the SMEX-Lite ACS software. Next, the context in which the software resides is explained. The paper describes the benefits of encapsulation, inheritance and polymorphism with respect to the implementation of an ACS software system. This paper will discuss the design of several software components that comprise the ACS software. Specifically, Object-Oriented designs are presented for sensor data processing, attitude control, attitude determination and failure detection. The paper addresses the benefits of the OOD versus a conventional procedural design. The final discussion in this paper will address the establishment of the ACS Foundation Class (AFC) Library. The AFC is a large software repository, requiring a minimal amount of code modifications to produce ACS software for future projects, saving production time and costs.