A Consolidated ACS Flight Software Development Approach for the Earth Observing-1 Spacecraft

Kathie Blackman and Jeff D'Agostino
the Hammers Company
7474 Greenway Center Drive, Suite 710
Greenbelt, Maryland 20770
301-345-5300
kblackman@hammers.com, jdagostino@hammers.com

Abstract. The Earth Observing 1 (EO-1) mission is part of NASA's New Millennium Program (NMP). The EO-1 Attitude Control System (ACS) flight software was based on the Tropical Rainfall Measuring Mission (TRMM) flight software, both of which were developed by the Hammers Company, Inc. Lessons learned during TRMM ACS software development led to a consolidated software development approach for the EO-1 ACS.

The approach started with a standalone system that incorporated the "first-cut" flight software into the spacecraft simulation, allowing closed loop simulations to run on a desktop computer. Consequently, more algorithm and coding errors were detected earlier in the development process. The consolidated system was connected directly to a ground support equipment computer in order to develop test procedures, ground system databases, and display pages. For flight software testing, the system's spacecraft simulation module was used as a spacecraft simulator, and the flight software was removed and loaded onto the test or flight hardware.

This type of consolidated development approach decreased dependency on hardware deliveries and allowed for a reduced level of software testing to continue through hardware down time. This approach helped meet the challenge of EO-1's shorter schedule and lower cost as compared to TRMM.

.

I – Introduction

EO-1 Overview

The Earth Observing-1 spacecraft (see Figure 1) is part of NASA's New Millennium Program (NMP), which was created to accelerate the development of new space-applicable technologies and to reduce risk and cost to future missions by conducting dedicated technology testing missions. The focus of EO-1 is to develop and test new remote sensing, spacecraft, and operations technologies for future land imaging missions[1]. The EO-1 spacecraft will be used to validate three advanced hyper-spectral imagers: the Advanced Land Imager (ALI), the Hyperion instrument, and the Atmospheric Corrector (AC). Additionally, EO-1 will fly seven new technologies: the X-Band Phased Array Antenna (XPAA), the Carbon-Carbon Radiator (CCR), the Lightweight Flexible Solar Array (LFSA), the Wideband Advanced Recorder Processor (WARP), the Pulsed Plasma Thruster (PPT), Enhanced Formation Flying (EFF), and the Fiber Optic Data Bus (FODB).

EO-1 Attitude Control Subsystem

The EO-1 Attitude Control Subsystem (ACS) is three-axis stabilized using three on-axis reaction wheels (RW) for control actuation and a hydrazine propulsion system (four thrusters) for ΔV capability. The EO-1 spacecraft also has three on-axis magnetic torquer bars (MTB) which are used for momentum management, and the experimental PPT that will be used for pitch attitude control[2].

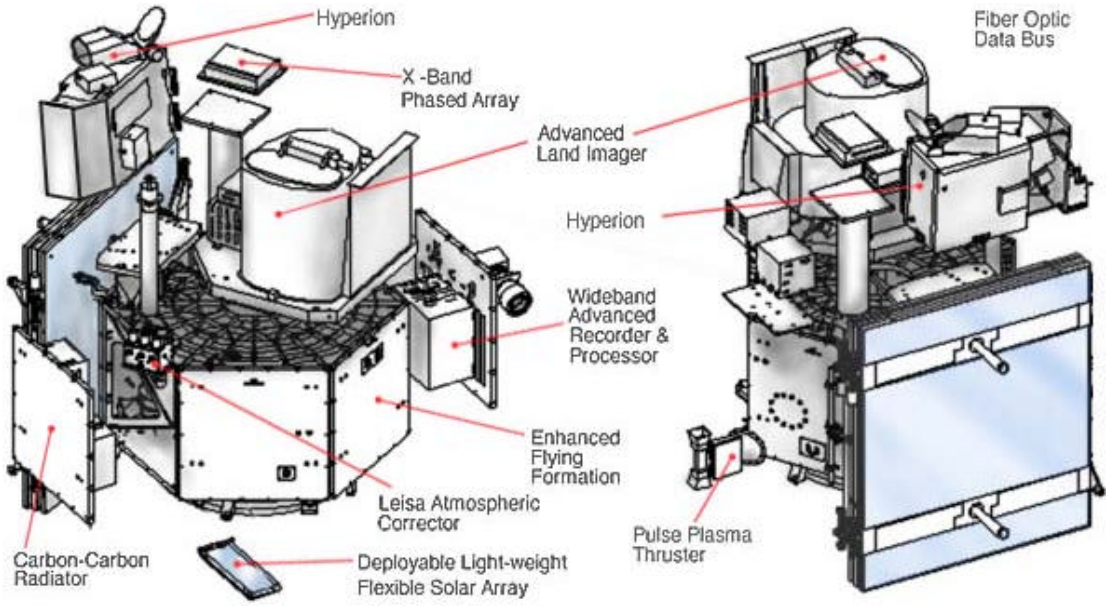

Figure 1: The EO-1 Spacecraft

The ACS sensor suite consists of four coarse sun sensors (CSS) which are used for initial sun acquisition, a three-axis Inertial Reference Unit (IRU) used for rate determination, a three-axis magnetometer (TAM) used for magnetic field measurements (and for rate damping when in BDot control), and an autonomous star tracker (AST) for attitude determination during normal mission operations.

The ACS computer system (see Figure 2) consists of a network of R000 processors acting as Remote Service Nodes (RSNs) on a 1773 spacecraft bus. Along with the RSNs is a R3000 architecture-based Mongoose V processor, which acts as the 1773 bus controller and also hosts the Command & Data Handling (C&DH) software and the ACS software. The Attitude Control Electronics (ACE) RSN provides the direct interfaces to the IRU, TAM, RW, MTB, PPT, and thrusters. The housekeeping (HSK) RSN manages the interface with the on-board Global Position System (GPS), and the XPAA RSN maintains the interface with the XPAA. The AST is directly connected to the 1773 bus with no RSN interface.

EO-1 ACS Software

The ACS software is responsible for reading and processing the ACS data from the ACE and HSK RSNs and the AST, determining the attitude of the spacecraft, and generating the proper commands for the actuators. The ACS software must track the spacecraft position and velocity (using both GPS data and an internal ephemeris propagator). It is also responsible for processing commands, generating telemetry, managing a variety of control modes, and monitoring the health of the ACS suite.

The EO-1 ACS software is designed to run as a single task on the Mongoose V processor. Written in C, it is managed as part of the suite of C&DH tasks, and has no direct access to external devices. That is, all external interfaces to the ACS software are managed through various C&DH function calls (see Figure 3). The ACS software accesses data and commands via function calls to the C&DH Software Bus task, and sends telemetry via function calls to the same task. Spacecraft time is received via function calls to the C&DH Time Code task, and the ACS software can issue ASCII string messages via calls to C&DH utility functions.

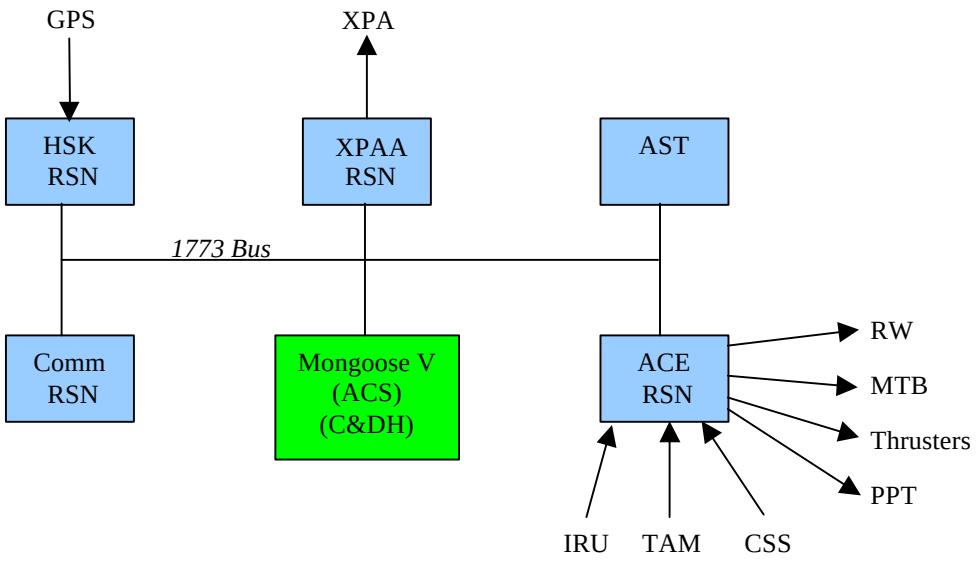

Figure 2: EO-1 Attitude Control Subsystem Computers

The EO-1 ACS software was derived from the ACS software developed for the Tropical Rainfall Measuring Mission (TRMM). Significant code re-use was initially planned, but design changes early in the life of the project required most of the code to be re-written to meet unique EO-1 needs. However, the software design philosophy of a modular approach to the individual components of the ACS software remained the same. The modular design of software similar to that of EO-1 has been covered elsewhere[3,4], and while it is not the focus of this presentation, it eased implementation of the consolidated development system discussed in Section III.

EO-1 ACS Flight Software Build History

The EO-1 ACS software had four major builds. ACS Build 1.0 was completed in May 1997 and implemented the initial ACS control algorithms. It was developed solely with the consolidated development system discussed in Section III. ACS Build 2.0 was completed in October 1997 and updated the control algorithms as well as the interfaces to the C&DH software. This was the first build to be integrated with the C&DH software in the lab environment. ACS Build 3.0 was completed in September 1998 and implemented interfaces for the newly added ACE Safehold controller. This build also closed out most problem reports generated during testing of Build 2.0. ACS Build 4.0, completed in March 1999, updated default parameter values and closed more problem reports generated during testing. At the time of this writing, the latest build is ACS Build 4.1, although one more build is expected to implement final updates for default parameter values.

II – Background (History of TRMM Development)

TRMM Overview

The Tropical Rainfall Measuring Mission (TRMM) was launched in November 1997. A joint mission between NASA and the Japanese National Space Development Agency (NASDA), TRMM is designed to monitor tropical rainfall and the associated energy that influences the atmospheric circulation that

affects the global climate[5]. TRMM is flying five instruments: the Precipitation Radar (PR), the TRMM Microwave Imager (TMI), the Visible and Infrared Scanner (VIRS), the Cloud and Earth Radiant Energy System (CERES), and the Lightning Imaging Sensor (LIS).

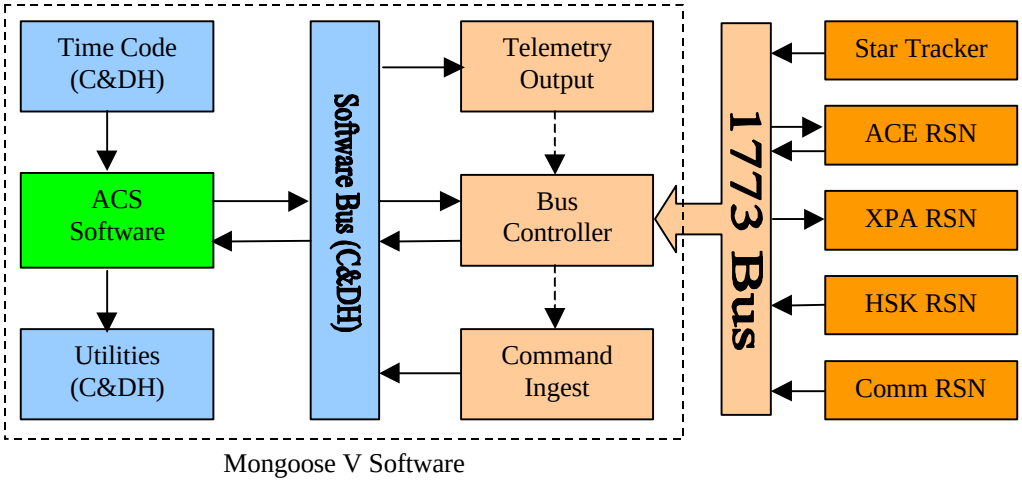

Figure 3: EO-1 ACS Software External Interfaces

The TRMM spacecraft consists of an ACS similar to that of EO-1, with the exception that TRMM is heavily redundant. The TRMM ACS suite includes two ACE computers (one acting as a hot backup), four reaction wheels, two sets of MTBs, two IRUs, two TAMs, eight CSSs (two sets of four), two Digital Sun Sensors (DSS), twelve thrusters, and an Earth Sensor (with a redundant data path). The ACS is also responsible for commanding the two Solar Array Drives (SAD) and the two High Gain Antennae (HGA) to the correct position[6].

The TRMM ACS software resides on an 80386 CISC processor (as opposed to EO-1's Mongoose V RISC processor) in a similar configuration as on EO-1. That is, all external interfaces to the ACS software are managed through C&DH function calls. As noted previously, the ACS software is modular in design.

TRMM Flight Software Development Environment

The TRMM ACS flight software was written in C, using desktop code editors. The code could not be compiled at the developers' desks. Instead, the source code had to be moved to the build PC in the ACS software development lab and compiled and linked there. This required that a C&DH build be ready for use by the ACS developers before they could build and test their code. It also required that the ACS software development lab be ready for use, which meant that all the components of the lab had to be completed and functioning properly for ACS software development to progress.

The ACS software development lab (see Figure 4) consisted of an 80386 ACS processor, a build PC, two ACEs, a spacecraft dynamic simulator on a VAX computer, a KineticSystems Computer Automated Measurement and Control (CAMAC) interface between the ACEs and the simulator, a Communications processor simulator, and a Ground Support Equipment (GSE) computer for running test procedures and monitoring the ACS software.

In this configuration, the flight software is on the ACS processor breadboard, and sends actuator commands to the ACE which are sent through the CAMAC to the actuator models residing in the dynamic simulator. Conversely, the sensor data generated in the dynamic simulator models are

passed out through the CAMAC to the ACE and then to the flight software on the breadboard. The dynamic simulator has a 1773 card, allowing it to simulate the Gimbal and Solar Array Control Electronics (GSACE) remote terminal to accept commands to the solar arrays and high gain antennae, and to generate position and rate measurements. The GSE computer is used to send commands to the flight software and to collect and display telemetry.

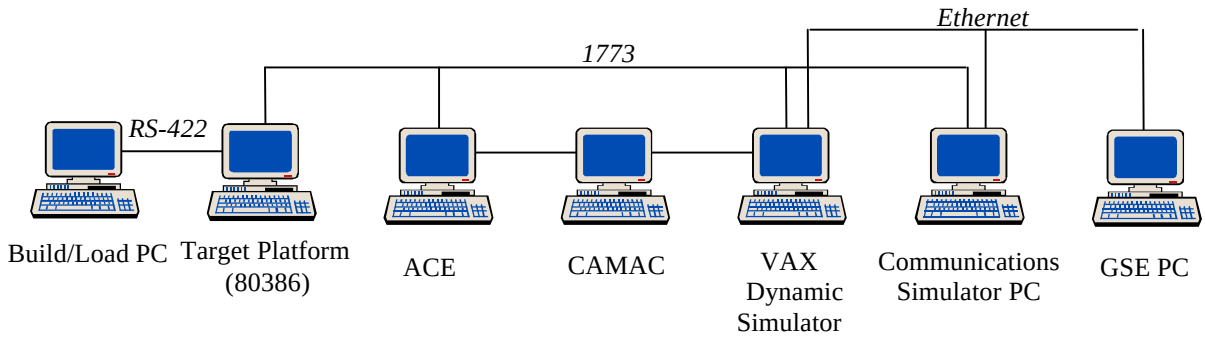

Figure 4: TRMM ACS Software Development Lab

TRMM Simulator Development

The TRMM attitude control analysts used a Fortran simulator on a VAX mainframe to test their control algorithms. It consisted of control algorithms that would later be used to develop TRMM ACS flight software, and a spacecraft simulator that included spacecraft dynamic models, sensor and actuator models, and ephemeris models. The simulator provided closed loop interaction between the control algorithms and the spacecraft simulator to support testing of the control algorithms.

Once the control analysts completed design of the control algorithms and the spacecraft simulator models, the control algorithms were written up in an algorithm document and delivered to the ACS software team for flight software development. The spacecraft simulator models were turned over to another independent contractor for development of the VAX dynamic simulator. The spacecraft simulator models were converted to C and ported to the VAX 4000/400 computer that supported hardware interfaces with the 1773 bus, the GSE computer and the ACE via the CAMAC.

After the spacecraft models were turned over for development of the dynamic simulator, the attitude control analysts continued to update and refine the models and the associated database values. It was a significant effort to ensure that the dynamic simulator models were kept up to date with the changes made by the analysts, and some problems that turned up during flight software testing were due to out-of-date models or database values.

TRMM ACS Software Testing

Prior to the completion of the software development lab, there was no capability to do any closed loop testing of the flight code and the simulator models in the VAX simulator. Unit testing could be done on individual sections of each system, but there was no interface between the two systems to allow them to pass data back and forth (See TRMM Unit Test in Figure 9). As a result, both the hardware interface and algorithm problems needed to be debugged at the same time once the software development lab was available. Any hardware delays in setting up the software development lab or any hardware breakage during the test process resulted in day for day software test delays.

Debugging in the software development lab proved to be time consuming since it required continuous addition of print statements, recompiling, and linking. Debugging the TRMM ACS software consisted of inserting debug ASCII messages in the source code which were output through a serial UART port back to the build/load PC. In this way, developers could see the execution sequence of the ACS code. However, if the developers wanted to change the location or contents of the messages, they had to edit the source code and rebuild and reload the software.

Flight software testing required the use of test procedures and command and telemetry databases that resided on the GSE computer. The procedures could not be dry run and the databases could not be checked out prior to the completion of the software development lab. Only a single set of components in software development lab was available for use by the flight software test team, the ACE software developers, the ACS software developers, and the VAX simulator developers. When any one of the groups was using the lab, all the other groups were prevented from performing any other closed loop testing

Although this process supported flight software testing of the highly successful TRMM spacecraft, the flight software test schedule was driven by the timeliness of hardware deliveries and hardware down time. To work in an environment requiring total dependence on hardware is not an ideal situation for a software engineer. For this reason, and because of the compressed schedule of EO-1, the Hammers Company (tHC) developed a consolidated software development approach for the EO-1 flight software.

III – Consolidated Development System

Consolidated Development System Overview

The consolidated development system (CDS) was created as an attempt to free the ACS flight software development process from dependencies beyond the control of the software developers. Initially, the EO-1 ACS software team considered using the VAX dynamic simulator from TRMM for its software lab. However, a launch delay on TRMM, coupled with discovered inconsistencies between the TRMM and EO-1 lab environments, led the team to realize that the effort to convert the TRMM system to EO-1 would be prohibitive for EO-1's development schedule.

The first successful run of the CDS prototype was in December 1996 as an after-hours experiment to execute the TRMM ACS flight source code on a desktop PC. Data were logged and graphed to show that the system successfully simulated a sun acquisition maneuver.

The EO-1 CDS was built in three different configurations – a standalone configuration for use by developers and analysts on their desktops; a GSE-interface configuration, which allowed a GSE computer to send ground commands and receive telemetry from a CDS computer; and a lab environment configuration, in which the ACS source code was removed from the CDS and integrated with the C&DH software for the target platform. In the lab environment configuration, the function of the CDS is the same as that of the dynamic simulator in the TRMM flight software development lab.

The three CDS configurations differ by compile options so that changes made to one of the configurations will automatically be applied to the others.

Consolidated Development System Design

The EO-1 CDS is a single executable program developed in C and C++ with the Microsoft Visual C++ Developer Studio for Windows. The main components of the CDS are the Main Interface, the ACS Thread and Models Component (see Figure 5).

The *Main interface* component of the CDS is designed to maintain communications between the ACS thread and models components, and to maintain communications with the outside world (via a GUI and network interfaces). The main interface is also responsible for maintaining the synchronization heartbeat between the components.

The *Models* component is the spacecraft dynamic simulator portion of the CDS. This component simulates the spacecraft orbit and attitude, accepts actuator commands from the ACS software via the main interface, and sends simulated sensor responses back to the ACS via the main interface. The models component is very similar in function to the TRMM spacecraft dynamic simulator, and can enable or disable a variety of environmental and hardware characteristics, such as aerodynamic drag, wheel friction, and sensor noise.

The *ACS thread* is the container for the ACS flight software source code. The source code for which the ACS development team is responsible contains no modifications unique to the CDS. That is, when code is updated and run under the CDS, it can be moved directly to the target platform development environment with no changes, reducing the risk of introducing errors in the code during the port to the software lab environment.

Since the ACS software is designed to run in an infinite loop (it enters a sleep state while waiting for new data), the source code must be placed in a separately executing thread in the Windows program to allow the rest of the CDS to execute.

The ACS source code externally references C&DH functions and sends and receives data packets to the outside world. In the CDS, the external references are simulated outside the ACS source code. Since these functions are not the responsibility of the ACS software developers, the developers of the functions must define the function formats, but the contents of the simulated functions may be tailored for the CDS environment. For the EO-1 CDS, the simulated C&DH functions call CDS library functions to perform data handling tasks and to check for the status of the environment. For example, a simulated software bus function could also check for ACS thread termination requests from the main interface.

The approach for the CDS was to integrate the ACS source code, with no modifications, into a single program with the spacecraft models. Special interface code was written to simulate the external interfaces to the ACS, such as CCSDS data packets, C&DH external variables and C&DH function calls. For the EO-1CDS , 3 CCSDS data packets, 7 C&DH external variables, and 10 C&DH functions were simulated.

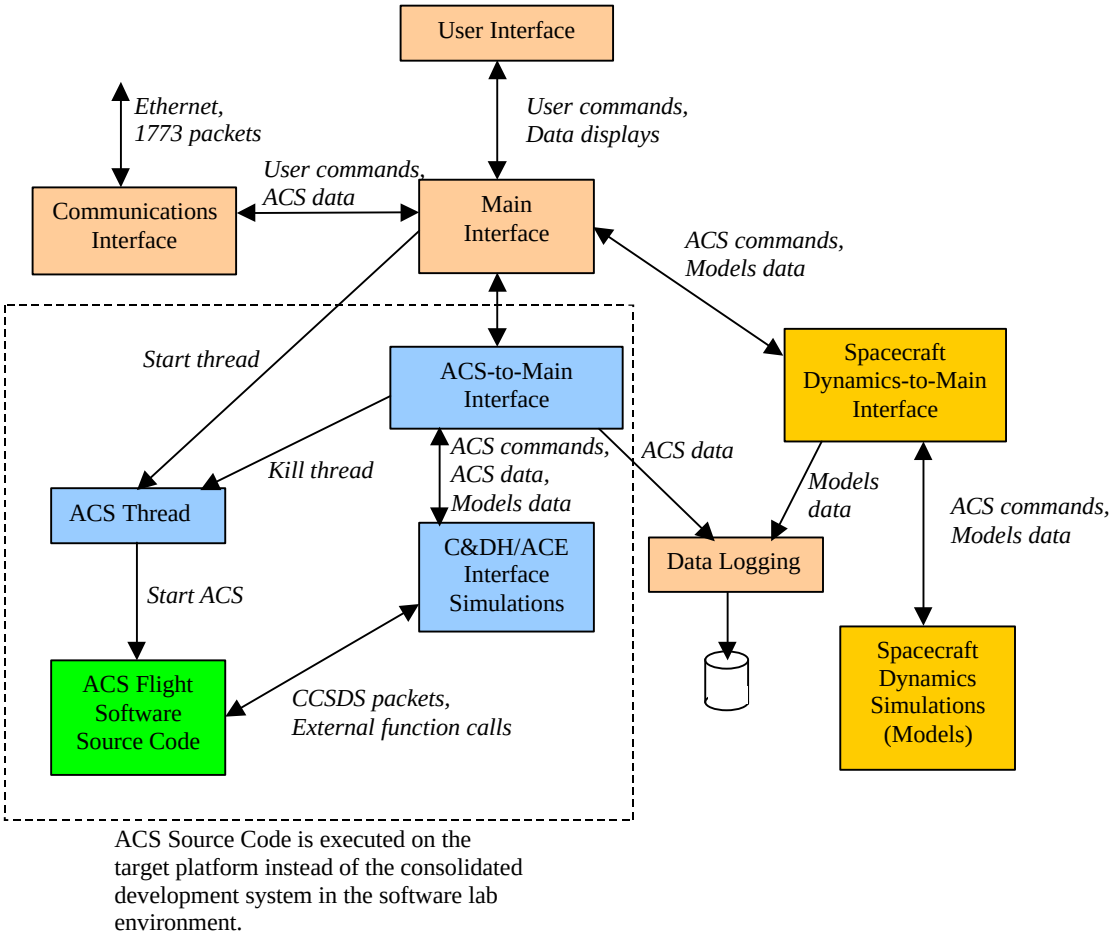

Figure 5: Consolidated Development System Design

Consolidated Development Configurations

The standalone configuration of the EO-1 CDS (see Figure 6) ran faster than real time and consisted of flight software source code and spacecraft simulator models. This configuration was used for flight software code development, spacecraft model development, unit testing by the software developers, and preliminary testing by the flight software testers. All that was needed to run the standalone CDS was an executable and input file so the simulation could be run on any desktop PC. The user interface allowed the user to send flight software commands, read in spacecraft model configuration files, and view and log telemetry.

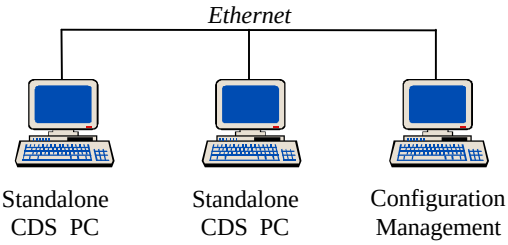

Figure 6: Standalone Configuration of Consolidated Development System

The standalone configuration allowed developers to pull code directly from configuration management over a local area network into the simulation/development environment, and allowed developers to share source files easily. Additionally,

debugging of the flight source code could be done at the developers' desks in the Microsoft Visual C++ environment with no need for a target platform.

The GSE-interface configuration (see Figure 7) ran in real time and was the standalone version of the CDS with an Ethernet interface component which transmitted and received CCSDS Standard Format Data Units (SFDU) to and from the GSE computer. The Ethernet connection supplied a command and telemetry link between the CDS and the GSE computer. With this interface, the flight software test procedures could be dry run and the command and telemetry databases could be checked out, all using the flight software source code that was running on the desktop PC. The only hardware that was required for this configuration was the desktop PC and the GSE computer. This version was used throughout the flight software test process as an alternative to the software development lab. Many tests were dry run or debugged on this system in order to allow more efficient use of test time in the lab. It was important to be able to continue testing without use of the lab because of hardware down time and because it was shared with the EFF software group.

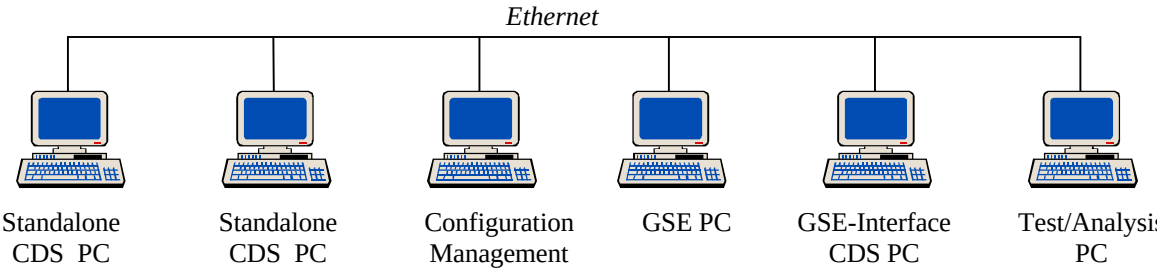

Figure 7: GSE-Interface Configuration of Consolidated Development System

The ACS software lab configuration of the CDS (see Figure 8) ran in real time and consisted of the spacecraft simulator models, a command and telemetry link to the ASIST, and a connection to the 1773 bus to simulate the ACE, AST and GPS remote terminals. Adding a 1773 interface component to the CDS allowed it to perform strictly as a spacecraft dynamic simulator and ACE simulator in the lab environment. Once the EO-1 target platform was ready, the ACS source code already had been heavily tested in the other configurations. In the lab environment the ACS source code was removed from the CDS and integrated with Command & Data Handling (C&DH) software for loading onto the target platform. Since the interfaces to the C&DH were simulated in the CDS, most of the integration was seamless. Test results from this environment were compared with test results from the GSE-only environment. When anomalies were found, scenarios were re-created at the stand-alone level, and developers could step through the code line-by-line to debug the problem. Using the stand-alone system to debug problems found in the lab environment freed up lab resources for additional testing and EFF software work.

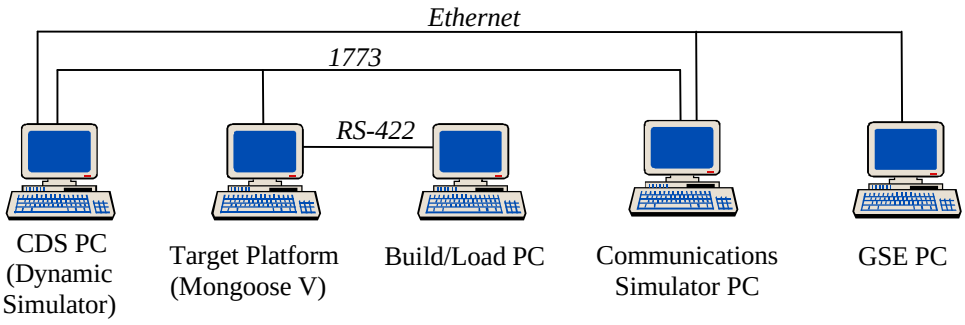

Figure 8: ACS Software Lab Configuration of Consolidated Development System

TRMM / EO-1 Schedule Comparison

The use of the CDS allowed EO-1 ACS flight software development to proceed despite the lack of a software development lab in the early stages of the development process. On TRMM, the ACS software development lab and C&DH Build 1.0 were ready for use in March 1993. The TRMM ACS Build 1.0 was delivered in August 1993 (five months after the lab was ready), and only contained basic interface code to the C&DH (no ACS control algorithms). The TRMM ACS Build 2.0, which included ACS control algorithms, was completed in April 1994 (thirteen months after the lab was ready).

By comparison, the EO-1 software development lab and C&DH Build 1.0 were ready for use in November 1997, while the EO-1 ACS Build 1.0, which contained most of the ACS control algorithms, was completed in May 1997 (six months earlier). The ACS test team was able to make use of the time between completion of the flight software delivery and the hardware delivery of the software development lab, which would have been forced idle time on TRMM, by starting to test the software using the GSE-interface CDS configuration. The EO-1 ACS Build 2.0, which included updates to the control algorithms as well as updates to the C&DH interfaces, was completed in October 1997, one month before the lab was ready for ACS integration. (Although some of the EO-1 ACS source code was inherited from the TRMM ACS, most of the software had to be re-written to meet unique EO-1 ACS requirements. Therefore, the EO-1 development schedule did not benefit as greatly from heritage code as originally expected.)

Advantages of the EO-1 CDS

Since the same version of spacecraft simulator models were used for both the standalone and the lab environment CDS configurations, the difficulties in maintaining two different systems as on TRMM were avoided. There was a Fortran PowerStation simulator with simulated flight software and spacecraft models used by control analysts early in the program that was a follow on from the system used by the TRMM analysts. This system was abandoned as soon as the initial version of the CDS standalone configuration was completed. Future desktop simulation work by the control analysts was accomplished using the standalone CDS system with the actual ACS source code since it was kept up to date with the latest software changes in support of software testing.

The CDS enabled EO-1 flight software developers to run closed-loop simulations at their desktops to help verify code before releasing it, and to assist in debugging of delivered code (See EO-1 Unit Test in Figure 9). In fact, one developer was able to continue writing and testing EO-1 ACS code on a notebook computer while in Japan supporting the TRMM launch.

Using the CDS, most of the flight software coding errors were found in the software debug environment so that once the ACS software was loaded on the target platform, only the interface issues still needed

to be worked out. The Microsoft Visual C++ development environment has an efficient debug system so that finding and fixing code errors was a much less time-consuming process than in the hardware environment on TRMM. Even when errors were encountered when running in the software development lab, the same scenarios could be set up on the standalone system to investigate the problems in a software debug environment.

The EO-1 ACS flight software test team was able to obtain the software lab GSE computer before delivery of the rest of the lab components. This allowed the test team to dry run all the flight software test procedures and check out the command and telemetry databases prior to running on the completed software development lab (See EO-1 Procedure Development in Figure 9). The test team was able to get a head start on closed loop testing and GSE procedure and database development that decreased the dependency on hardware deliveries.

The test team was able to borrow an additional GSE computer and use the GSE-interface configuration of the CDS to continue debugging test procedures without using valuable lab resources. This provided a "free" redundant test environment that doubled the available test time and allowed the Enhanced Formation Flying software development team to spend more time in the software lab without interference from the ACS group.

The portability of the standalone configuration of the CDS with the flight software and the spacecraft simulator allowed for wider distribution of the system. In the early stages of the project there was a request from the EO-1 Flight Dynamics group for a copy of the standalone CDS, which they then used to develop and test of the ground software used for telemetry processing and sensor calibration.

The mission operations team was provided with several GSE computers but not a breadboard on which to run the flight software. Using the GSE-interface configuration of the CDS, they were able to gain experience in sending ACS commands and validating ACS telemetry through the same GSE computer they will use to command the spacecraft during the mission.

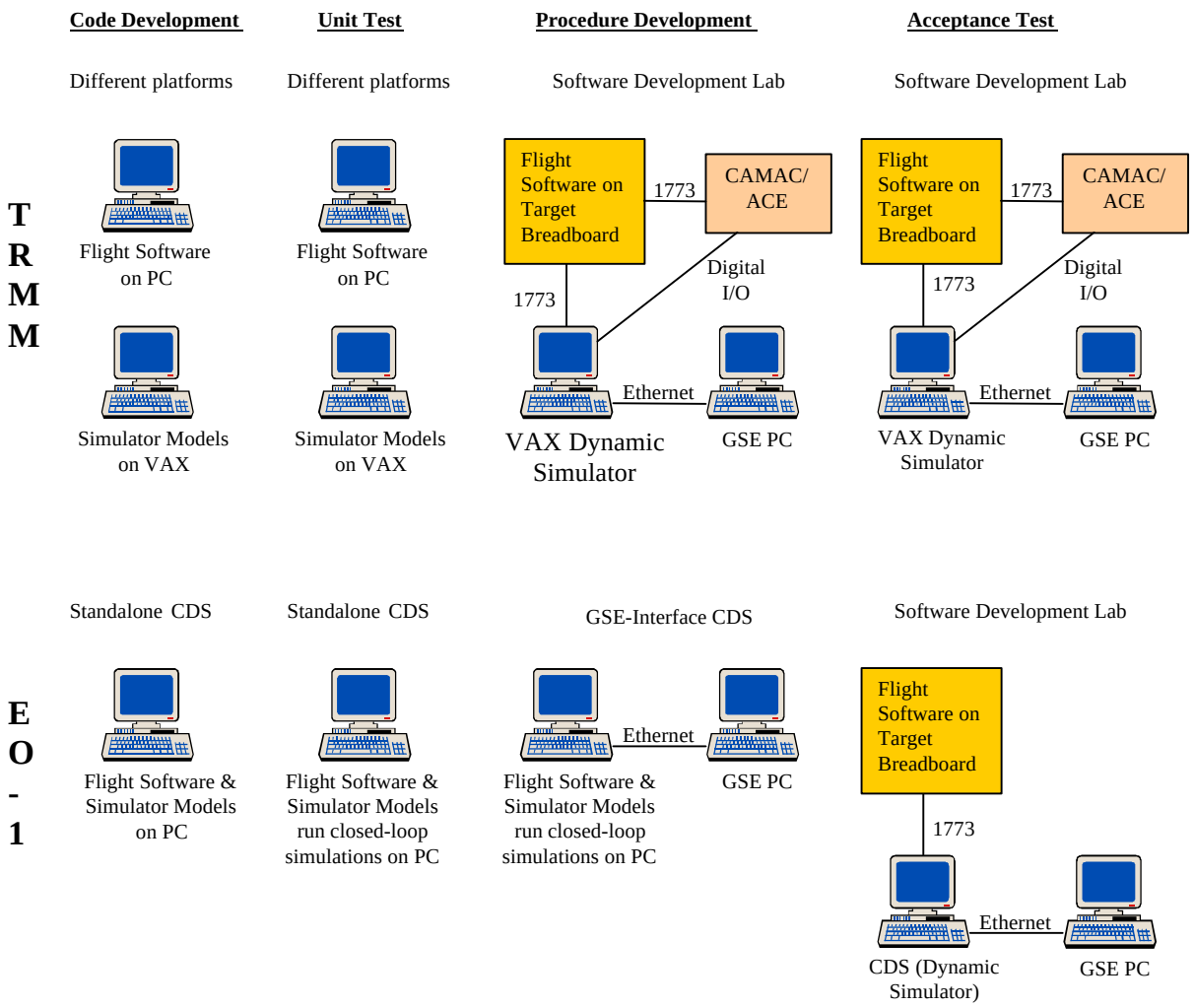

Figure 9: Comparison of TRMM and EO-1 Software Development Processes

Disadvantages of the EO-1 CDS

A disadvantage of this EO-1 software lab configuration of the CDS for acceptance testing in comparison to the TRMM system is that there was no hardware interface from the CDS to the ACE, so hardware-in-the-loop testing with the ACE as part of the configuration was not possible. The ACE was simulated as part of the CDS, even in the lab environment configuration. The ACE I/O and ACE safehold software testing was done by stimulating the sensors and measuring the actuator response in an open loop fashion.

A disadvantage of the GSE-interface configuration of the CDS for EO-1 mission operations training is that it did not simulate the C&DH software that resides in the Mongoose V spacecraft processor with the ACS software. Functions such as table loads and absolute/relative time command sequences that are often used by the mission operations team when the spacecraft is on orbit were unavailable in the CDS.

The standalone and GSE-interface configurations of the CDS did not allow developers to catch hardware-specific anomalies. For example, an unknown characteristic of the target platform caused a math exception that could not be duplicated in the CDS. This anomaly had to be researched, corrected, and verified in the lab environment.

Since the CDS was built using Microsoft Visual C++ on a Windows platform, it did not use the target platform's C make file. Therefore, problems with the make file (e.g., memory allocation errors) were only caught in the lab environment.

Future Plans

The consolidated development system created for EO-1 has proven successful enough to be used in new ACS development projects such as the the Naval EarthMap Observer (NEMO) and the Formation Flying Test Bed (FFTB). Additionally, plans are underway to expand and improve the design of the CDS to allow more uses of the system.

Among the future uses and improvements for the CDS are:

- Extrapolation a single spacecraft simulator into a collection of satellites to simulate a satellite formation.
- Addition of advanced data display and logging capabilities.
- Addition of a component development kit which would allow users to create and customize components without re-building the entire CDS.
- Addition of portions of C&DH flight software (from the SMEX-Lite project) into the CDS as separate components.
- Addition of a component which would allow the CDS to connect to third-party analysis tools (such as MatLab®) to test control algorithms before writing ACS code.
- Configuration of the CDS to allow for hardware-in-the-loop testing, which would enable the CDS to access hardware registers to read actuator inputs and stimulate sensor outputs.

IV – Conclusion

In the days of longer schedules and larger budgets, it was acceptable to deliver software that had only undergone basic functional and unit testing with the knowledge that future testing in a hardware environment would uncover more subtle errors in control algorithms, software design and coding. With shorter schedules and smaller budgets, this luxury is gone. Software must be more thoroughly tested at early stages of development for schedules to be met.

The consolidated software development system used for the EO-1 project enabled the ACS flight software developers to test and verify code despite delays in hardware deliveries. The minimization of downtime due to hardware delays prevented software delivery dates from slipping in the schedule. In fact, the first build of the EO-1 ACS was completed six months before any hardware was ready to run it, and the CDS allowed testing of the software to proceed during that six-month period.

Besides aiding the software coding portion of ACS software development, the CDS facilitated the development of telemetry and command databases and GSE procedures needed for formal testing of the ACS software in the software lab environment. Additionally, once the ACS flight software reached the lab environment for integration on the target platform, the CDS served as a dynamic simulator. Using the same simulator throughout the development process reduced the risk of errors being introduced due to differences in simulations.

Since the consolidated development system was a small program (which fit on a single floppy disk for EO-1), it could be easily distributed to analysts and mission operations personnel throughout the development cycle. This allowed members of the EO-1 team to gain an early familiarity with the ACS flight software before delivery of the code on the target platform. It also allowed more people to catch potential errors in the code that could then be corrected before formal delivery of the next build (more eyes are more likely to find errors). This reduced the workload on formal testing, as errors

that normally would have not been caught until formal testing already had been corrected.

In the standard development approach, design, coding and testing are treated as independent and nearly separate phases of the development process, making redesigns or code changes more costly to implement. The consolidated development system encouraged a rapid approach to ACS software development, since designs were updated and tested more easily, and coding and testing took place concurrently. This approach reduced overall development time.

The CDS does not eliminate the need for a target platform and a software development lab. For instance, processor-unique errors (byte-ordering, exception handling, etc.), target make file errors (memory allocations, etc.), and external function incompatibilities (ill-defined interfaces, etc.) will not necessarily get caught in the CDS environment. However, by testing the ACS source code to debug algorithm-level problems, the CDS allows the lab environment to be used more efficiently for these types of errors. The ACS software is therefore more likely to meet the demands of today's "faster, better, cheaper" spacecraft environment.

V – Acknowledgments

The authors wish to thank all the managers, engineers, developers and testers who contributed greatly to the development and improvements of the consolidated development system and to the development of the EO-1 and TRMM ACS flight software, especially: Louise Bashar, John Bristow, Stephan Hammers, Albin Hawkins, Teresa Hunt, David Kobe, Kequan Luu, Steve Mann, Paul Sanneman and Seth Shulman

VI – Acronym List

ACE	Attitude Control Electronics
AC	Atmospheric Corrector
ACS	Attitude Control Subsystem
ALI	Advanced Land Imager
AST	Autonomous Star Tracker
C&DH	Command and Data Handling
CAMAC	Computer Automated Measurement and Control
CCSDS	Consultative Committee for Space Data Systems
CDS	Consolidated Development System
CERES	Cloud and Earth Radiant Energy System
CSS	Coarse Sun Sensor
Comm	Communications
DSS	Digital Sun Sensor
EFF	Enhanced Formation Flying
EO-1	Earth Observing-1
FFTB	Formation Flying Test Bed
GPS	Global Positioning System
GSE	Ground Support Equipment
HGA	High Gain Antenna
HSK	Housekeeping
IRU	Inertial Reference Unit
LIS	Lightning Imaging Sensor
MTB	Magnetic Torquer Bar
NASA	National Aeronautics and Space Administration
NASDA	National Space Development Agency (Japan)
NEMO	Naval EarthMap Observer
NMP	New Millennium Program
PPT	Pulsed Plasma Thruster
PR	Precipitation Radar
RSN	Remote Services Node
RW	Reaction Wheel
SAD	Solar Array Drive
SFDU	Standard Format Data Unit
SMEX	Small Explorer
TAM	Three-Axis Magnetometer
tHC	the Hammers Company
TMI	TRMM Microwave Imager
TRMM	Tropical Rainfall Measuring Mission
UART	Universal Asynchronous Receiver/Transmitter
VIRS	Visible and Infrared Scanner
XPAA	X-Band Phased Array Antenna

VII – References

1. N. Speciale, "Earth Observing-1", May 10, 1999, http://eo1.gsfc.nasa.gov/miscPages/home.html (July 13, 1999)

2. P. Sanneman, K. Blackman, M. Gonzalez and D. Speer, "New Millennium Earth Orbiter-1 Mission: Attitude Control Requirements and Capabilities", 21st Annual AAS Guidance and Control Conference, February 1998 (AAS98-002)

3. K. Barnes, C. Melhorn and T. Phillips, "The Software Design for the Wide-Field Infrared Explorer Attitude Control System", 12th Annual AIAA/USU Conference on Small Satellites, August 1998 (SSC98-VII-2)

4. S. Andrews and J. D'Agostino, "Development, Implementation, and Testing of the TRMM Kalman Filter", 1997 Flight Mechanics Symposium, May 1997

5. TRMM Web Site – NASA, "A Global Eye on Tropical Rainfall", http://trmm.gsfc.nasa.gov (July 13, 1999)

6. the Hammers Company, Inc. (prepared for NASA GSFC), "Tropical Rainfall Measuring Mission Attitude Control System Software Requirements Specification", March 16, 1995 (TRMM-ACS-SRS-CD-R03.0-031695)

VIII – Author Biographies

Kathie Blackman graduated from Lehigh University (BSME) in 1990 and Purdue University (MSME) in 1992. Ms. Blackman has developed spacecraft dynamic simulators for the X-Ray Timing Experiment (XTE) and the Tropical Rainfall Measuring Mission (TRMM) in support of ACS flight software testing. Since 1996 she has worked for the Hammers Company, Inc., on EO-1 as an ACS controls analyst and flight software test lead.

Jeff D'Agostino graduated from Virginia Polytechnic Institute and State University in 1984 with a BS in Physics and a BA in Philosophy. He has worked in the NASA Goddard Space Flight Center environment since then, including work as an ACS flight software engineer since 1987. Mr. D'Agostino has developed ground operations software the Extreme Ultraviolet Explorer (EUVE), and has developed, tested and/or maintained flight software for the International Ultraviolet Explorer (IUE), the Solar Maximum Mission (SMM), EUVE, TRMM, EO-1, and the Naval EarthMap Observer (NEMO).