COTSAT
Small Spacecraft Cost Optimization for Government and Commercial Use

Aaron J. Swank

David Bui, Christopher Dallara, Shakib Ghassemieh, James Hanratty
Evan Jackson, Pete Klupar, Michael Lindsay, Kuok Ling, Nicholas Mattei,
David Mayer, Emmett Quigley, Stevan Spremo[*], Zion Young

NASA Ames Research Center, Moffett Field, CA 94035, USA

Cost Optimized Test of Spacecraft Avionics and Technologies (COTSAT-1) is an ongoing spacecraft research and development project at NASA Ames Research Center (ARC). The prototype spacecraft, also known as CheapSat, is the first of what could potentially be a series of rapidly produced low-cost spacecraft. The COTSAT-1 team is committed to realizing the challenging goal of building a fully functional spacecraft for $500K parts and $2.0M labor. The project's efforts have resulted in significant accomplishments within the scope of a limited budget and schedule. Completion and delivery of the flight hardware to the Engineering Directorate at NASA Ames occurred in February 2009 and a cost effective qualification program is currently under study. The COTSAT-1 spacecraft is now located at NASA Ames Research Center and is awaiting a cost effective launch opportunity. This paper highlights the advancements of the COTSAT-1 spacecraft cost reduction techniques.

I. Introduction

Cost Optimized Test of Spacecraft Avionics and Technologies, COTSAT-1, is a rapid prototype, low-cost spacecraft for science experiments and technology demonstration. The spacecraft platform is designed to accommodate low-cost access to space for variable remote-sensing payloads, while maintaining an architecture allowing future expansion for potential biological payloads. COTSAT-1 is a $\sim 400\,\mathrm{kg}$ small spacecraft that is baselined to accommodate a remote sensing instrument as the primary payload. The primary objective for a COTSAT-1 mission is to rapidly deploy a spacecraft with a minimum six month reliable-performance period. The goal of COTSAT-1 as a technology demonstration unit is to demonstrate the ability for drastic cost reduction in spacecraft design and to develop methods and technologies for maximizing reuse of developed spacecraft hardware, software and related technology on future missions. The developmental cost of the spacecraft platform is significantly reduced by housing the bus and payload subsystems in a single-atmosphere, artificial environment. This concept has been proven with the first man-made satellite, Sputnik, developed by the former Soviet Union.[6] Other Soviet sealed environment spacecraft have been designed and flown successfully. This heritage design attracted the interest of ARC because of dramatic cost reduction possibilities.

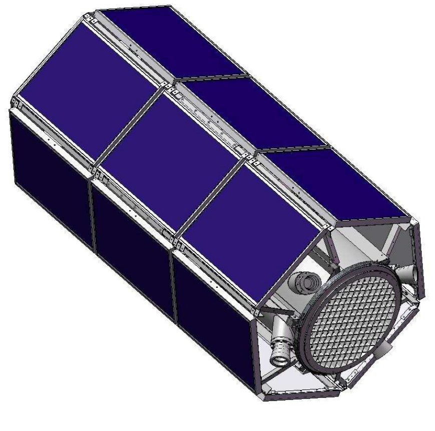
Figure 1. COTSAT Solid Model

[*]Corresponding author.

II. Cost Reduction and Technology Reuse

Many of the advancements in overall cost reduction for COTSAT-1 can be attributed to the use of the one-atmosphere pressurized structure to house spacecraft components. For example, the artificial environment makes it readily feasible for incorporating a wide array of pre-built hardware, including Commercial Off The Shelf (COTS), Modified Off The Shelf (MOTS) and Government Off The Shelf (GOTS) hardware. Hybridizing a one-atmosphere pressure vessel with current COTS technologies allows for subsystem cost reduction, in most cases, by orders of magnitude.[3] By using COTS hardware, the spacecraft program can utilize investments already made by commercial vendors. Indeed, COTSAT-1 makes extensive use of COTS, MOTS, and GOTS hardware to save on development time and cost. Where possible, COTSAT-1 also incorporates industry data interface standards such as USB 2.0 and Ethernet for reduced development and integration time of the COTS components. When inexpensive COTS hardware solutions are not readily available, subsystems are designed and developed in-house at ARC. In-house developed hardware for COTSAT-1 includes for example the reaction wheels, star tracker and electrical power systems.

Proper design of the software architecture for COTSAT-1 has additionally proved to generate significant cost reduction through simplified development and integration time. As noted by Wilmot:[7] "an obvious way to reduce cost and schedule is to increase the amount of software reuse." Indeed, the COTSAT-1 project utilizes GNU/Linux and open source software for reduced development time and increased code reuse. The amount of software reuse is increased on COTSAT-1 by leveraging off of existing software libraries and device drivers already written for COTS electronics. In some cases, minor modifications to the open source or industry supplied software drivers were necessary to simplify integration or to add desired features. However this approach has still proven to be effective and significant in cost reduction, compared to a completely new development and testing effort.

III. Structure

A key design characteristic of COTSAT-1 for cost reduction and reduced development time is the single atmosphere, artificial environment which encompasses the vast majority of the satellite components. An artificial environment container, which comprises much of the satellite structure, is used to contain the single atmosphere environment. Refer to Figure 2 for a graphical depiction of the structure comprising the artificial environment container. During integration, the container is filled with one-atmosphere standard laboratory air. The artificial atmosphere container is used to replicate an Earth-like atmosphere, allowing the use of COTS hardware and electronics which were not necessarily originally designed to operate in the vacuum environment of space. The artificial atmosphere environment additionally allows for heat transfer within a convective medium aiding in heat distribution. The air within the container aids in transferring heat from hot electronics or heaters to cooler portions of the spacecraft. Two fans move air within the container to promote an isothermal gaseous environment for electronics, extending overall spacecraft subsystem lifetime.

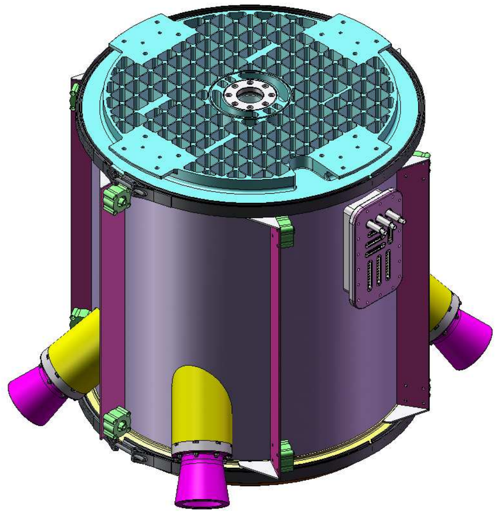
Figure 2. Artificial Atmosphere Container

The sealed container is comprised of an aluminum cylinder with monolithic aluminum end-plates at each end. The cylindrical portion of the sealed container is primarily fabricated from rolled-and-welded sheet aluminum and features longitudinal ribs and billet flanges. These ribs help prevent structural failure in modes including axial loading, acoustics, and buckling; additionally, these ribs provide attachment points for an otherwise void round surface. Both end-plates feature an external waffle grid pattern to provide a stiff, yet lightweight structure capable of heat dissipation for any thermally sensitive components. On the reverse side, a grid of blind threaded holes is provided to allow for non-specific, modular avionics equipment buildup within the container. The aft end-plate is designed to mate with the launch vehicle separation system.

IV. Electrical Power Subsystem

The COTSAT-1 Electrical Power System (EPS) architecture utilizes a distributed power and self-monitor approach. The distributed power and self-monitor approach is adapted from GeneSat-1 and allows for the reuse of all systems for future missions without major redevelopment.[2] The distributed power architecture provides methods for mitigating Single Event Upsets (SEUs) or latch-ups without compromising the entire system. The highly configurable system contains two major elements: the Central Power Controller (CPC) and up to 32 Remote Power Controllers (RPC's). The CPC contains a microprocessor to perform centralized power control for the entire spacecraft and additionally monitors system health. The CPC receives voltage and current monitoring signals from each RPC allowing the CPC to implement active software control for detection and mitigation of SEUs or latch-ups experienced by on-orbit radiation. Each RPC is responsible for regulating voltage and safely limiting current from an unregulated power source. Current monitoring circuitry on the RPC will automatically disable the power regulator on the RPC should a latch-up incident, failure, short or open circuit occur. After a preconfigured delay, the circuitry resets the device and attempts to restore power. The architecture allows for dual string redundancy for mitigation, isolation, and recovery from component failure. Additional details on the CPC and RPC architecture are found in Spremo et al.[4,3]

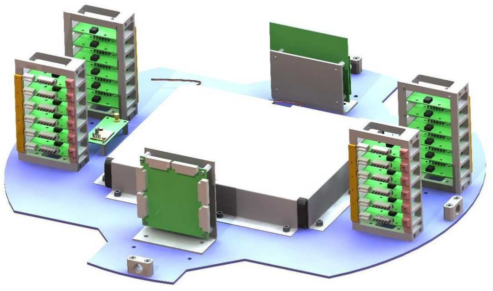
Figure 3. Electrical Power System

V. Command and Data Handling

The command and data handling (C&DH) subsystem provides a number of critical capabilities, including spacecraft health and status monitoring, communication, payload science data management and subsystem management. The C&DH subsystem for COTSAT-1 uses a two-tiered system consisting of the CPC and a PC-104 computer. The CPC is the system master controller and is designed to perform a limited set of essential tasks with simplicity and robustness in mind. In addition to basic power management capabilities, the CPC handles essential survival functions of the satellite, including system health, problem mitigation and communication. The CPC is responsible for primary command and data handling for communication with the ground station. The CPC interfaces with a PC-104 computer via an RS232 communication link and utilizes this channel to relay commands and system status information. This interface allows for commands to be sent, processed, and executed by the PC-104 system. The PC-104 computer handles all complex tasks such as star tracker attitude estimation, attitude control system tasks, payload image processing, payload data storage, and high data rate down-link communication.

The C&DH architecture for the PC-104 utilizes standard Ethernet UDP with acknowledgement or TCP/IP to communicate between the flight computer executive and each of the essential satellite subsystems. Subsystem communication functions as an on-board subscribe & publish communications network. Any subsystem which requires spacecraft attitude information need only to subscribe to the information provided. From subsystem to hardware, any number of protocols may be used, depending on the designed hardware interface, for example USB, Ethernet, TTL, RS232 or RS422. Although the the actual hardware device may be connected via different communications protocols, for example a serial communications link, the PC-104 flight software treats each subsystem as a device connected via Ethernet and a software daemon is responsible for translating data packets to the proper protocol required for the destination device. For COTSAT-1 hardware, USB and Ethernet interfaces were preferred due to available hot-plugging capabilities and ease of integration. The Ethernet architecture for command and data handling provides a number of benefits in the system design. For example, the Ethernet architecture generates a design capable of easy subsystem reuse, thereby providing cost reduction for a series of missions. The Ethernet architecture also has the benefit of allowing for combined simulation/hardware in the loop testing and verification. With a Ethernet connection, the various satellite subsystems may be running on various distributed systems connected only by a network. In one instance a software developer was testing control algorithm routines on a local desktop workstation in one room, while accessing actual hardware sensors on the development test-bed in another room. This methodology aided in parallel software development and testing as well as verified the capability of remotely running simple development tests without physically entering a clean room facility, where the flight hardware would be located. The Ethernet-based architecture therefore provides the foundation for the ultimate FlatSat, namely a distributed network of hardware devices, subsystem assemblies, or even inexpensive desktop workstations. For COTSAT-1, hardware simulators were used to aid in parallel software development. These hardware simulators were software daemons on the PC-104 or a desktop workstation, which receives and responds to Ethernet packets. Having designed the subsystem interfaces to support Ethernet, it was easy to observe system response to anomalies or abnormal behavior both with and without actual hardware present. Simply by creating the proper Ethernet data packet, one could inject any desired hardware response to the system being tested. As a result, the Ethernet architecture allowed for rapid development and verification of desired performance characteristics.

COTSAT-1 utilizes USB 2.0 to interface with a number of the satellite hardware devices, including for example the IMU and star trackers. One benefit of utilizing USB, is that the udev system available on GNU/Linux provides a foundation for hot swappable or plug and play hardware. Udev is a device manager daemon providing for dynamic device addition and removal to the system. The developer may also write udev rules for the system to perform when a new device is initialized or removed from the system. The COTSAT-1 software developers have utilized the udev system for identifying and hot-plugging hardware. For example, a udev rule was written for the USB IMU and star trackers. Upon recognition of the device to the operating system, the hardware manufacturer and serial number was identified via udev to determine the proper associated hardware configuration parameters specific to the satellite layout. Similarly, based on the hot-plugging of devices and recognition via udev, the system utilizes the available resources. For instance, there are multiple star tracker units present in the system as well as two IMU's. In the case of the star trackers, the subsystem default behavior is to utilize the maximum number of available star trackers for attitude determination. Alternatively, the subsystem may be instructed to utilize a specified number of star trackers to reduce overall system load. When attitude information is queried from the star tracker subsystem, the software intelligently combines the most recent set of quaternion estimates from the available star trackers to determine the satellite attitude. In the case of the IMU, the system defaults to the first available IMU device, yet the information may also be requested from a specific hardware device.

COTSAT-1 utilizes IP addressable radios for primary bidirectional data communication via the PC-104 flight computer. Data transfer from the communications subsystem to the communications radio is therefore performed via an Ethernet link. Sending data packets to any communications radio, or even a ground station, is a simple matter of rerouting Ethernet packets to a different destination address/port. It is worth noting that the Ethernet interface also supports hot-plugging of devices. For COTSAT-1 Ethernet devices such as the communication radio or primary satellite subsystems, the system has been constructed primarily with static device addresses and port numbers for simplicity. Hot-plugging of Ethernet capable devices is easily implemented using MAC addresses and dynamic host configuration protocol (DHCP), and is therefore considered an option for future modular reuse of hardware devices.

V.A. Communication Architecture

The COTSAT-1 communications architecture incorporates four independent communications paths. The CPC is coupled with one bidirectional communications link and a beacon in the amateur radio band for public outreach purposes. The PC-104 flight computer is outfitted with one primary bidirectional communications link and a secondary unidirectional communications link for high speed data downlink capabilities. The software architecture has been designed to accommodate any number of additional communications links through the PC-104 flight computer interfaces. The primary spacecraft command and control, communication, as well as spacecraft health and status telemetry is handled via the bidirectional communications link on the CPC. The unidirectional, high speed communications downlink on the PC-104 is for payload data.

The flight software on the PC-104 is responsible for routing data command and control packets to the appropriate Ethernet port which may be the CPC or primary spacecraft subsystems. The communications link on the PC-104 is therefore capable of routing command and control information from the PC-104 flight computer or communications radio to the CPC and vice versa, providing a backup communications link. The bidirectional link on the PC-104 is also responsible for providing additional detailed spacecraft health and status telemetry to the ground station.

VI. Software Architecture

In addition to immediate development and cost reduction, a desired goal for the COTSAT-1 software architecture was to promote and maximize potential software reuse. This was achieved with COTSAT-1 by generating a set of core software daemons for common spacecraft functions. On the PC-104, the software architecture consists of modular, independent software daemons for each subsystem or capability. A software daemon exists for the star tracker, the IMU, the reaction wheels, the main executive, the communications system, the control system and the payload. Each software daemon is given a pre-determined priority level for the multitasking operating system. Major subsystem software daemons communicate via UDP/TCP. The modular software architecture allows for parallel development and testing before integration. In addition, the architecture allows for subsystem reuse on future missions.

In addition to utilizing separate software daemons for major subsystems, the COTSAT-1 software also makes extensive use of dynamically loadable shared object libraries. Each module, for example the module to interface with hardware such as the IMU, is compiled into a dynamic shared object library. The shared object libraries not only adhere to the modular design, but also provide flexibility in software upgrades and simulation. In the case of COTSAT-1, the host operating system was the same as the target PC-104. As a result, the flight software shared object libraries could then be used in Matlab/Simulink for simulation purposes. The procedure allows for verification of the flight software functionality, for debugging purposes and also allows for testing system response with simulated environmental conditions.

The operating system selected for COTSAT-1 is Debian GNU/Linux with the Linux kernel compiled with Ingo Molnar's low-latency kernel preemption patch and high resolution timers to support real-time performance. The Debian GNU/Linux distribution was selected due to the vast number of available software packages and the ease of streamlining the system to a minimal set of installed components. In addition, by selecting GNU/Linux as the satellite operating system, COTSAT-1 was able to leverage upon a number of industry or open source community supported COTS hardware. Open source software libraries and community supported software programs were also leveraged to further reduce development costs. For example, as already described, the software architecture consists of modular software daemons for major satellite subsystems. COTSAT-1 utilizes an open source software utility called Monit to monitor and manage the satellite software daemons. The Monit daemon monitoring tools are used to ensure the software daemons are in nominal operating conditions. If a subsystem daemon is no longer functioning properly, the monitoring daemon is configured among many options to automatically restart the subsystem or to notify the PC executive.

Revision control, configuration management and coordination between software developers on multiple test platforms is achieved by use of Subversion and the Debian package management system. Each software daemon and shared object library for the satellite software is distributed as a Debian package. Software installation via Debian packages ensures proper software version dependency resolution and proper previous software version removal. The software packages are distributed to the flight hardware and development/test platforms using a networked server running the Advanced Packaging Tool, APT. APT is a Debian package management utility which additionally handles dependency resolution. Upon a request to install a particular software package, APT searches for dependencies and then downloads and installs or updates the software packages as necessary. For example, the entire software suite for COTSAT-1 is installed from the remote APT server by simply issuing the command to install the "cotsat" package. The procedure allows efficient and cost effective development of the flight software as well as an ability to reuse software packages across various spacecraft missions. By using the power of the Debian package management system, parallel rapid development/testing and configuration management is achieved.

In order to increase potential code reuse, standard open source GNU tools for software development are utilized for COTSAT-1. Wilmot[7] notes that the adoption of standard open source GNU tools aids in establishing commonality between build tools and reducing dependency on proprietary tools, thereby increasing the potential software reuse and hence cost reduction. For example, task scheduling and prioritization on COTSAT-1 is implemented using Posix Threads (PThreads), a standard threading library. By using a POSIX compliant threading structure, the software will operate on any POSIX compliant system, preventing the software from becoming locked to a particular design. Additionally, the use of a standard library maximizes reusability and the generated code is more readily understood by other software developers, hence reducing training time and cost.

VII. Attitude Determination and Control

COTSAT-1 has a three-axis Attitude Determination And Control System (ADACS), using four reaction wheels and three magnetic torque coils. Nominal operations include the use of the reaction wheel system for three-axis pointing control with the torque coils used for wheel desaturation. The torque coils are also essential for initial stabilization of the satellite after launch and for detumbling the spacecraft when the PC-104 system is not fully initialized. The torque coils may also be used by the PC-104 flight computer to produce additional torques should the reaction wheel system become degraded during nominal operations. Both the reaction wheels and the torque coils are developed in-house at NASA Ames. Developing the reaction wheel system at ARC has generated wheels at 10% of the cost for similar wheels, while still achieving the same performance characteristics and durability.

The control system requires input from the attitude determination sensors and the reaction wheel system (control system actuators). Consistent with the software design architecture, each subsystem communicates with resource subsystems via Ethernet packets. The interface between the control system and the actuators is a data packet with a set of requested body torques. The control system actuator subsystem daemon transforms the requested body torques into individual low-level control commands which are sent to the actuator hardware. In the case of COTSAT-1, the requested torques are translated into voltage control signals for each reaction wheel.

The information exchanged between the control system and the attitude determination sensors is a quaternion describing the satellite attitude and satellite body rotational rates. The control system need not necessarily have information as to the actual source of the determined attitude information. Although the concept was conceived independently by the COTSAT-1 project, the design philosophy is similar to that of Graven et al.,[5] where the nature of the information exchanged stems directly from the physics rather than from the hardware devices present. The attitude information knowledge is abstracted from the actual source device. The attitude control system simply assumes that the satellite has the capability of providing attitude information and rotational body rates at a minimum desired rate. For COTSAT-1 the attitude information is provided via an attitude estimator subsystem daemon. The COTSAT-1 attitude estimator is an extended Kalman filter, which utilizes body rate sensor information and regular star tracker quaternion updates. A USB MicroStrain IMU is used for body rate information. Although the MicroStrain IMU contains an integrated magnetometer for absolute orientation determination, magnetic fields generated from satellite components such as the reaction wheels is expected to interfere with magnetometer readings. As a result, the attitude control system for COTSAT-1 only utilizes the IMU for body rate information.

Since the control system software daemon is separated from the attitude estimator subsystem, a change in the method to determine spacecraft attitude information will not require a change to the control system software modules. For instance, the attitude estimator could receive information from a different combination of attitude sensors such as a sun sensor, horizon sensor, or an IMU, yet the same basic attitude information in the form of a quaternion and body rotational rates would be supplied to the control system. By designing the system to accept a standard set of parameters related to the physics rather than directly from hardware sensor and actuator inputs/outputs, the ability to mix and match various hardware devices to provide the same capability is maximized. Similarly, the ability to reuse developed technology and subsystem modules for future missions is enhanced.

VII.A. Star Tracker

Most star trackers utilize extremely customized and proprietary designs down to the circuitry level, from which much of their extremely high cost is derived. However, it is possible to use existing off-the-shelf technology to create a star tracker with competing performance to its custom counterparts. By using a COTS imaging device, removing software processing from the star tracker itself, and developing software in-house, cost was greatly reduced and the ability to reuse technology for future missions was greatly enhanced. For example, since the camera and lens are COTS products, the hardware can be easily interchanged with other hardware to suit various mission requirements. The software would then only require configuration file modifications for the new hardware, such as to identify the orientation and location of the device within the satellite. The same star tracker software can be applied to different hardware with very little additional development. The system is also fully expandable, such that adding units for redundancy only incurs additional hardware costs.

For COTSAT-1, there are four star trackers on the satellite, each of which is comprised of a Lumenera monochrome machine-vision camera and a FUJINON lens. The star tracker cameras are all connected to and powered by the PC/104 stack via USB 2.0 ports. Image processing and attitude determination is performed on the PC/104 stack. In order to identify stars in images, the software contains a star database derived from the 118,218-star Hipparcos catalog.[1] The database contains a list of every star pair within the camera field of view and the angular distance between those pairs. It also contains the inertial position information for each individual star directly from the Hipparcos catalog. The spacecraft attitude information is computed directly as a quaternion from the star field image information.

For rapid prototyping, the star tracker quaternion attitude estimation algorithm was originally written entirely in Matlab/Simulink and then integrated with the rest of the system via Real-Time Workshop auto-code generation. The use of Simulink/Real-Time Workshop auto-code generation has promoted rapid prototyping capabilities and then where necessary the Simulink models are supplemented with hand written C/C++ code. The entire star tracker software and estimation algorithms on COTSAT-1 is now entirely written in native C++ with the GNU tool chain for enhanced performance. For simulation purposes, the native C++ software is included into the Simulink models by using the shared object libraries compiled for the star tracker system. Simulink model blocks which had previously represented the star tracker prototype model are therefore directly replaced with the external library. The use of the GNU tool chain removes the dependency on proprietary, hardware specific libraries. In addition, the flight software is more readily utilized on other platforms and operating systems. The switch of the software design from Real-Time Workshop auto-code generation to native C++ for the star tracker brought about a number of additional advantages, including: 1. Simplified integration, 2. Improved configuration management capability via Subversion, 3. Eliminated proprietary software dependencies and licenses, 4. Increased portability to other hardware, 5. Maximized software modularity and the potential for reuse, and 6. Better performance.

VIII. Test Platform

To aid in technology development and testing, the COTSAT-1 hardware and technology performance has been verified by a number of prototype test-beds. There have been three major test platforms during the development cycle.

The first test platform consisted of a single degree of freedom system with one reaction wheel. For simplicity the first platform utilized an aircraft grade IMU (not suitable for space applications) and a single board computer running Microsoft Windows. A single star tracker was also integrated into the system. For rapid prototyping purposes, a simplified event driven state machine for the system was executed via Matlab/Stateflow on a network connected desktop workstation. The first prototype demonstrated functionality of custom-built hardware, electronics, and software for items such as the reaction wheel, the star tracker, the CPC and RPC. Resulting performance and knowledge gained by initial integration aided in directing the future development of both the hardware and architecture.

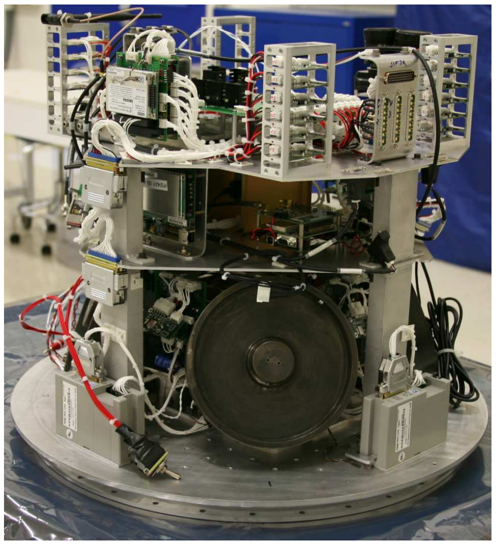
Figure 4. Third Test Platform

The second development test platform incorporated additional flight prototype hardware, providing a platform for in-the-loop testing of flight engineering models. The second platform incorporated four reaction wheels in the flight configuration, two star trackers, and the flight IMU for testing of the reaction wheel control system. The system utilized the flight PC-104 computer running Debian GNU/Linux. The second platform was the first to begin using the actual flight software, even though only the main control system and related subsystems had been tested and completed to flight specifications. A basic PC executive to operate the system was written in C/C++ using PThreads and was executed directly on the PC-104. Initially, the second prototype platform did not utilize the flight communications radios, but was later outfitted to utilize the flight bidirectional radio on the CPC and the IP addressable bidirectional radio on the PC-104 .

The third test platform, Figure 4, is a proto-flight unit, consisting primarily of flight versions of hardware and software. Proto-flight hardware on the third test platform includes for example: reaction wheels, four star trackers, two redundant IMU's, lithium-ion batteries, payload camera, PC-104 flight computer with heat pipe thermal solution, magnetometer (not present in previous test platforms). The final flight revision of the custom-built CPC and RPC circuitry is also present and the entire satellite is wired with the flight wiring configuration to test for unexpected noise or grounding issues.

The complete flight software architecture is implemented on the third test platform. Software modules which have been verified during test procedures of the previous test platform have been established as flight ready. Knowledge gained during testing of the control system on the previous demonstration unit aided to complete the integration of the attitude control system sensors and actuators. The flight version of the star tracker software, and attitude estimation techniques is implemented, as well as a number of modules to control basic devices such as the IMU and reaction wheel. The third test platform is the first to implement the flight version of the main PC executive and radio communications routines. The final flight versions of the software have yet to be completely verified on the proto-flight unit, but are expected to require only minimal modification to achieve flight status. It is expected that some configuration parameters will need to be modified, such as control loop gains to reflect the actual final satellite mass properties.

IX. Final Remarks

The Cost Optimized Test of Spacecraft Avionics and Technologies project, COTSAT-1, has examined and demonstrated a number of cost reduction techniques for rapid spacecraft design. As presented in this paper, COTSAT-1 has adopted a number of design philosophies and industry technologies not currently widely accepted within the spacecraft design community. One of the primary enabling technologies is the adoption of the artificial environment container, which further provides for a number of cost reduction techniques, such as the use of COTS technologies. Open source software further expands the cost reduction techniques by promoting a wide array of software reuse and design architectures. Furthermore, adopting widely accepted standard interfaces such as Ethernet and USB promotes cost reduction by increasing the number of available off the shelf hardware and software solutions. Selecting widely adopted standard data interfaces additionally enables accelerated testing, prototyping and parallel development without necessarily having access to the host platform.

References

[1]ESA, 1997, The Hipparcos and Tycho Catalogues, ESA SP-1200.

[2]E.P. Lee et al. The *.sat CubeSat bus: When three cubes meet. In *19th Annual AIAA/USU Conference on Small Satellites*, 2005.

[3]S. Spremo et al. Low-cost rapid response spacecraft, LCRRS. A case study in small satellite cost optimization for government and commerical use. In *Proceedings of the AIAA Space 2008 Conference*, San Diego, CA, 2008. SSC08-IV-7.

[4]S. Spremo et al. Low-cost rapid response spacecraft, LCRRS (CheapSat) - A research project in low-cost spacecraft design and fabrication while maintaining flight standards in a rapid prototyping environment. In *22nd Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 2008. SSC08-II-4.

[5]P. Graven, Y. Plam, L.J. Hansen, and S. Harvey. Implementing plug-and-play ADCS to support operationally responisve space. IEEEAC Paper No. 1586, December 2007. Version 2.

[6]G.S. Vetrov. *S.P. Korolev i ego delo: svet i teni v istorii kosmonavtiki.* Nauka, Moscow, 1998. (in Russian).

[7]J. Wilmot. Implications of responsive space on the flight software architecture. In *Proceedings of the AIAA 4th Responsive Space Conference*, Los Angeles, CA, April 2006.