QuakeSat Lessons Learned:
Notes from the Development of a Triple CubeSat

Contributors
(alphabetically):

Tom Bleier
Paul Clarke
Jamie Cutler
Louis DeMartini
Clark Dunson
Scott Flagg
Allen Lorenz
Eric Tapio

This report of QuakeSat lessons learned is a compendium of notes and recommendations of what to do (and what not to do) when building a nanosatellite or CubeSat. There are certainly many differences between building a nanosatellite and a large satellite, and they are highlighted in this report. Nevertheless, knowledge and experience gained from building large satellites is useful when building nanosatellites. That knowledge is also reflected here.

This project was performed on a shoestring and against a challenging time schedule. The entire project (through launch) was completed in 18 months; the launch date was unslippable (Eurokot does not slip); obtaining export licenses was time-consuming and, apparently, a first for a U.S. satellite on a Russian launcher. Read the following in that context, and recognizing that the satellite was finished on time, launched, and operated successfully for many months.

Table of Contents

1 Early Planning and Requirements

1.1 Decision to Build a Satellite

The first thing that a team or person should do is to decide if they really want to build a particular satellite. If you have a "customer" who has a mission that can only be accomplished by a satellite, it may be an easy decision.

Once a decision to proceed is made, and a basic "mission" is stated and agreed to, then there should be some time spent on goals for the mission. These goals should be listed and prioritized since time and funding will rarely allow you to complete and fly the spacecraft the way you originally had planned. Some features must be sacrificed if funding and time become scarce.

1.2 Assessing Cost

If cost is the driver, then a detailed cost breakdown should be done to look at all the costs: design, construction, integration, test, launch, early orbit, on-orbit operations, data analysis, and any additional science analyses that may be required.

The cost of QuakeSat (and the 3 engineering models built) in parts alone was $50K. The cost of this simple triple CubeSat launch was only $120K. The "people cost" to design, build, test, launch and fly the satellite—and to analyze the science data was between $1-2M. Cost growth can happen very quietly, and some thought should be given to what features or objectives that could be "eliminated" if costs rise, or alternatively, what other sources of funding can be accessed to cover the costs.

1.3 Requirements

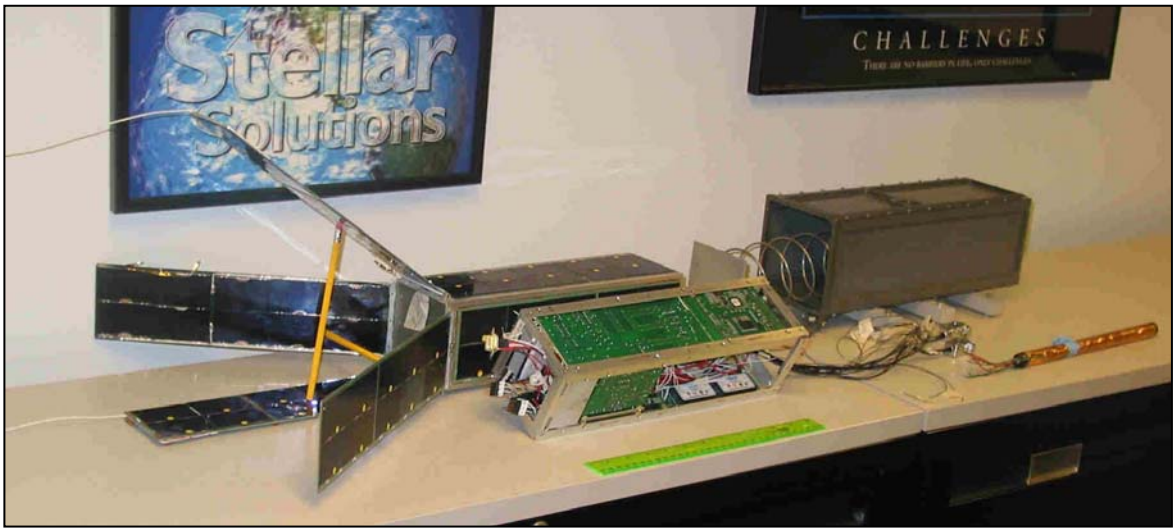
The project should have a complete requirements document. This may be a single page if the satellite is simple or many chapters if the satellite is complex and there are many interfaces.

There should be a formal System Requirements Review (SRR) and it may be prudent to have a "grey beard" review of the requirements. An experienced outsider can spot disconnects between the requirements and the funding and time allowed for completion. The comparison should be done in the context of how the satellite is being built and the processes involved. (A student-built satellite may have more relaxed requirements than a commercially-built satellite.)

2 Satellite

At the highest level, power and communications are the most important elements to mission success. Without them you have no mission; with them you can make something of even a damaged or failing spacecraft.

Below is a list of the major components used in QuakeSat. These parts have now been flown and have worked for 11 mos (at this writing), and with the exceptions noted, they have performed very well.



QuakeSat exploded for display in our conference room

Mfg	Part No.	Component	Notes
Rose Electronics	LI-2S1P	7.4 V, 1.5 AHr Li-Ion Battery	Worked fine for 7 mos. Should be potted with RTV to prevent electrolyte evaporation
Maxim	Max 1873	Li-Ion Battery Charge Controller	OK but noisy
Maxim	MAX 471	Current monitor chip	Works well (pos. only)
National	LM-34	Temp sensor def F (-55 deg to 185 degF)	Worked well
Maxim	Max4678	Low power switch	sensitive to latch up
Fairchild	NDP 6020P, FDP 6030L	High Power switch	May need heat sink
PowerOne	DFC6U5S5	DC-to-DC power	High ripple-add filter

Mfg	Part No.	Component	Notes
		converter	
Diamond Systems	Prometheus	66MHz CPU with 16-bit 16-channel A-D	Worked well. Removed Ethernet chip, clock batt, reduce 100 MHz to 66 MHz to reduce power
ICOM	TEKK 9600	UHF radio with AX.25 packets (half duplex)	Replaced electrolytic caps with tantalums. Added conductive foam to pwr transistor (heat xfer). Installed custom crystal for 436.675MHz frequency
Tigertronics	BayPAK9600	modem	worked well
?	CM8870PI	DTMF chip for watchdog	worked well
Motorola	PIC 16F628-20P	PIC processor for watchdog	worked well
National	74C00N	Quad NAND for watchdog	worked well
TI	2N2222	NPN Transistor for watchdog	worked well

2.1 Communications

Although we had problems and issues came up with our ground stations and communications, in general, our communications was robust. We understood our limited baud rate during the initial design. More bandwidth would have been better because it was the major factor limiting the amount of mission data we could collect. However, in general the overall reliability of our communications system allowed us easy and consistent access to the spacecraft and allowed us to have the mission we had. A number of other small satellites that flew on the same booster never established communications or had very poor communications which was a major limiting factor to their mission success.

It was very important to have a beacon to help find and identify the satellite, especially as our satellite was ejected among many other small satellites. The beacon should be strong (QuakeSat had 1 watt output at 436.675 MHz), long enough and often enough to easily detect the satellite (QuakeSat was 1 packet—a 200-byte payload— with spacecraft identifier, and repeated every 10 seconds).

2.1.1 Onboard Modem

Many spacecraft use an onboard modem to communicate with the ground. Modems are notorious sources of noise, often swamping out other signals. QuakeSat had a sensitive magnetometer, so internal noise was to be avoided. Since they are a part of the communications system, their effect may not be seen readily as they may be powered off when performing ground testing. They often operate by interrupting the CPU, and often they do so with a lower priority interrupt, causing jitter in the timing of their servicing routine being called. They also can be guilty of sending the interrupt over a parallel cable, thus slamming the system with EMI noise at frequencies like 600 Hz. Several spacecraft have discovered this amazing phenomenon once on-orbit.

2.1.2 Link Utilization

Find ways to better use the uplink/downlink. Here are some possible ways to do this for QuakeSat.

(a) Use error detection and correction in the send/receive protocols. The way QuakeSat uses AX.25, a single bit error means the loss of a full packet (200 bytes of data). This is inefficient.

(b) Use compression that is optimized for the data being transmitted. Although QuakeSat uses bzip to compress our digitized files, it does not perform very well. Other customized algorithms should perform better (e.g. by using delta modulation, non-linear compression, or some other signal processing technique before normal data compression).

(c) Plan heavier use of on-board (up-loadable) scripts throughout the mission.

(d) A software modem, on both the satellite and on the ground, should produce better performance (lower bit error rate) than the hardware TNC that was used.

(e) If a bit error is detected (a bad packet), try flipping every bit to see if we can find the one bit that is in error. This saves re-sending a packet when there is only one bad bit. This technique trades link bandwidth for processing power on the ground.

2.1.3 Communication Software

The satellite communication software was developed independently from the payload control software. The communication software was built in three parts: a send program, a receive program, and a driver used by the send program, the receive program, and the command/control software. In fact, the satellite command and control software could only receive and transmit individual commands. It did not have file transfer capability even though one of the satellite missions was to download 3 megabits of data per day. This allowed the communication development, testing, and fine-tuning to be performed independently of the satellite payload software.

The communication software had two features that mitigate the 9600-baud limit of the communication pipe between the satellite and the ground station. First, the communication software validated the checksum of each packet it received. A command was not executed by the satellite unless it was received correctly from the ground. This eliminated the concern that a partial command was executed, an important point when executing operating system commands (like file deletion) where an incorrect or partial command can have devastating effects. Second, none of the packets sent by the satellite had to be acknowledged. The HAM radio connection is generally slow and can have bit errors. When a file transfer was started the satellite would continuously transmit data until it reached the end of the file. The receiving ground station would then request again any packets with errors. Even with this communication mechanism a normal satellite contact would yield less than forty percent of the theoretical throughput. Contact time was spent doing activities other than downloads, such as establishing ground/satellite communication and getting satellite status.

2.2 Attitude Determination and Control

2.2.1 Magnetic Stabilization

Pre-launch analysis predicted that our mass properties were not sufficient to maintain a gravity gradient attitude profile. Consequently, passive magnetic attitude stabilization was chosen. The design was for the spacecraft to fly with its long axis parallel to the earth's magnetic field lines. In this plan the spacecraft would have rotated about the long axis twice per orbit. Around the poles we expected it might lag from this attitude as the field lines change more rapidly there.

However, this was apparently not the attitude our spacecraft ended up flying. Loss of the solar array current measurements severely limited our ability to determine attitude. The satellite's attitude profile has only been defined in general terms by using an infrared sensor that was pointed outward along the magnetometer boom, and could detect the bright earth or the darkness of space. The IR sensor indicates that our spacecraft may have been generally nadir pointing, in a gravity gradient type manner. A general nutation about the long axis of the spacecraft with a period of approximately 4 to 5 minutes has been measured consistently throughout the mission. The cone angle of this nodding is variable with respect to nadir, likely because the spacecraft's magnets interact with the earth's magnetic field (a planned result, but not sufficient to support the attitude profile predicted at launch). The angle is likely below 60 degrees, and could be substantially below 60 degrees. Indications of a spacecraft roll rate of about once every 15 to 20 minutes has been detected. This was not fully investigated and was only noticeable during the short periods the spacecraft's beta angle was close to 90 degrees. This profile has been fairly consistent throughout the mission, with the possible exception that the nutation angle may have decreased slightly during the mission.

2.2.2 Attitude Prediction/Determination

It was intended that the individual solar panel current profiles would be used to help determine the attitude of the spacecraft, knowing the local sun vector as the satellite flies around the earth. However, the solar array currents were lost by the very early loss of multiplexer-2.

With the solar panel currents unavailable, the wide-field IR sensor, a 12-panel solar array bus voltage, and a boom temperature were the only remaining attitude-varying telemetry from the spacecraft. They were sampled at as high a rate as we could support (data downlink limited), but were probably not sufficient for the task.

The use of the tiny IR detector was originally planned only as a binary measure of whether we were pointed up or down (earthward or spaceward). However, with the loss of the solar panel currents it was pressed into service as our primary attitude sensor. It was valuable but insufficient for this task. Several sensors with overlapping fields of views or completely orthogonal to each other would have been very helpful here.

Under the circumstances, we needed more time, data, and tools to get a better understanding of the satellite's attitude. Better understanding of the IR sensor field of view and signal calibration, better understanding of the changes to bus voltage that were attitude related, better understanding of the magnetic interaction of our on-board magnets and the earth's magnetic field, better mass properties data, and better analysis tools were all needed to better resolve our attitude knowledge. Time and money were limiting facts here.

2.3 Computer and A/D Converter

2.3.1 Choosing Hardware and Software

2.3.1.1 Computer Selection

We chose a general purpose CPU:

- PRO: Ease of design; can support an operating system
- CON: Power hungry 2.5 W

CPU selection is an important aspect of any successful small satellite development effort. The current marketplace provides a wide variety of commercial off-the-shelf (COTS) hardware and software. The use of COTS hardware and software reduces cost, design time, and possibly the size and weight of the CPU system. An additional benefit is increased performance. The competition in the CPU marketplace allows the satellite architect to select a hardware/software package that can be adapted to any payload requirement.

Current state of the art CPU systems include integrated data acquisition, communication, power supply circuitry, and storage subsystems to name a few. Some designs are modular. For instance, the I/O subsystem that supports Ethernet connectivity on the Quakefinder CPU was used for software development, but was not included on the operational satellite where it would be dead weight. The Quakefinder team selected the Diamond Systems Prometheus CPU system which included a built-in 16-channel, 16-bit A/D converter. It provided a rich but rugged operational system combined with a tunable, flexible, yet standard development environment.

Another aspect of the CPU system selection that cannot be overlooked was the availability and readiness of the CPU vendor to modify the base CPU system for the specific needs of the space mission. It was determined that the satellite computer should run at a slower clock rate than the commercial product. Diamond Systems made the required modifications in a timely and cost effective manner. This aspect of the CPU selection process must be managed based on market conditions and willingness of the vendor to overlook the lack of sales volume of a small satellite project.

2.3.1.2 Operating System Selection

We chose the Linux operating system for use on the satellite. Linux is a flexible Unix–like operating system that can run on a host of platforms from small embedded computers (1–2 MB with no memory management unit) to super computers.

Operating system (OS) selection is a pivotal aspect of the satellite development and operational environments. The choices range from a homegrown kernel to Windows CE. While the OS selection can take a back seat to the CPU selection, the payload development process will benefit if they are selected with each other in mind. This may mean compromises, but overall system stability should result if the two systems are compatible. A higher level of success is achieved if the hardware vendor supplies the OS. This was the case with the QuakeSat CPU vendor. Specifically, Diamond Systems sold a fully compatible Linux OS with their Prometheus CPU board. This meant that no time was wasted finding or developing I/O drivers, and that building an operating system for QuakeSat was as simple as burning the vendor's executable image into flash memory.

Linux was selected for a number of reasons relating to the development environment, the operational environment, third party software, and the compatibility with the ground station and satellite contact software. The selection ultimately proved successful for many of the same reasons that any imbedded project benefits from open source code. Linux provided a platform that allowed the operational software to be built and tested on the specific test environment and any generic Linux-based workstation. This supported a distributed development environment that required a minimum overhead to set up and support. It also gave the developers a common set of development tools and a vast working, documented, and tested code base that could be harvested over the Internet for free.

Another advantage was that several key device drivers were already written for AX.25 (packet protocol) and the Baypac modem.

Linux also provides a host of tools that were utilized in the operational satellite without burdening the software development team with coding, testing, and maintenance overhead. These include free utilities that report memory use, disk space utilization, software status, and date/time manipulation. The utilities were easily integrated into a set of diagnostic scripts that were used to report status to the contact software. Other useful utilities included compression (bzip2), data integrity checking (md5sum), and shell utilities like cat, grep, bash, etc.

Another benefit of any commercial operating system like Linux that supports an IP protocol stack is the existence of a ready-made communication path that exists below the application level software. Utilities like PING, SSH, and SCP provide an out of the box communication mechanism that supports an effective backup resource to the primary communication software. At Quakefinder we used the PING and SSH utilities to communicate with the satellite when the payload software temporarily failed. This helped the ground team validate modem and radio hardware health when normal satellite operations were impaired.

All of this meant that less then 10,000 lines of flight code needed to be written (3000 were device drivers).

The major disadvantages of using an OS like Linux are that (a) the great flexibility requires much more testing (there can be a high number of corner cases), and (b) there is limited

time-tagging accuracy (although our chosen Diamond Systems Prometheus computer addressed this with hardware timers).

There is another downside to using an off the shelf kernel like Linux for satellites that have a prolonged life span. Guaranteeing the validity of the Linux operating system on the spacecraft proved to be a task that was never completed. Fortunately this was never an issue. With any craft that operates in a hostile environment there comes the risk of software corruption. This can be caused by operational anomalies or environmental factors like high-energy particles corrupting flash memory. The problem is that a kernel like Linux requires the support of many files that work together and must be present in a valid state. With such a small team Quakefinder never had the time or developed the code necessary to validate all of the Linux files necessary to operate the payload software. It should be pointed out that Linux tools exist to solve this problem; one only needs time.

2.3.1.3 Software Development Environment

Equally important as the mission hardware and software supported by the CPU system is the development tools provided by the CPU vendor. A straightforward development environment combined with industry standard tools allows the satellite builder to draw from larger pools of software and hardware developers. This will also reduce development costs, or in the case of a university effort, create an appealing project. In either case the developers will recognize that the skill developed on the project will be useful in their future.

The Linux software development environment is an easy to learn set of operating system commands, tools, and practices. Almost all software engineers developed code with UNIX at a university or professional level. This makes software development straightforward. There are a number of universal, free, and reliable compilers and utilities available that support code development for the small satellite developer. Many large commercial organizations have adopted the Linux or UNIX environment for embedded software development. A small satellite software developer can leverage the power of a Linux development environment for less than $50 per developer. This is because the basic development tools are bundled with the operating system. Quakefinder standardized on the Red Hat version of Linux. Red Hat 9 is a reliable and robust platform.

2.3.2 Satellite Software

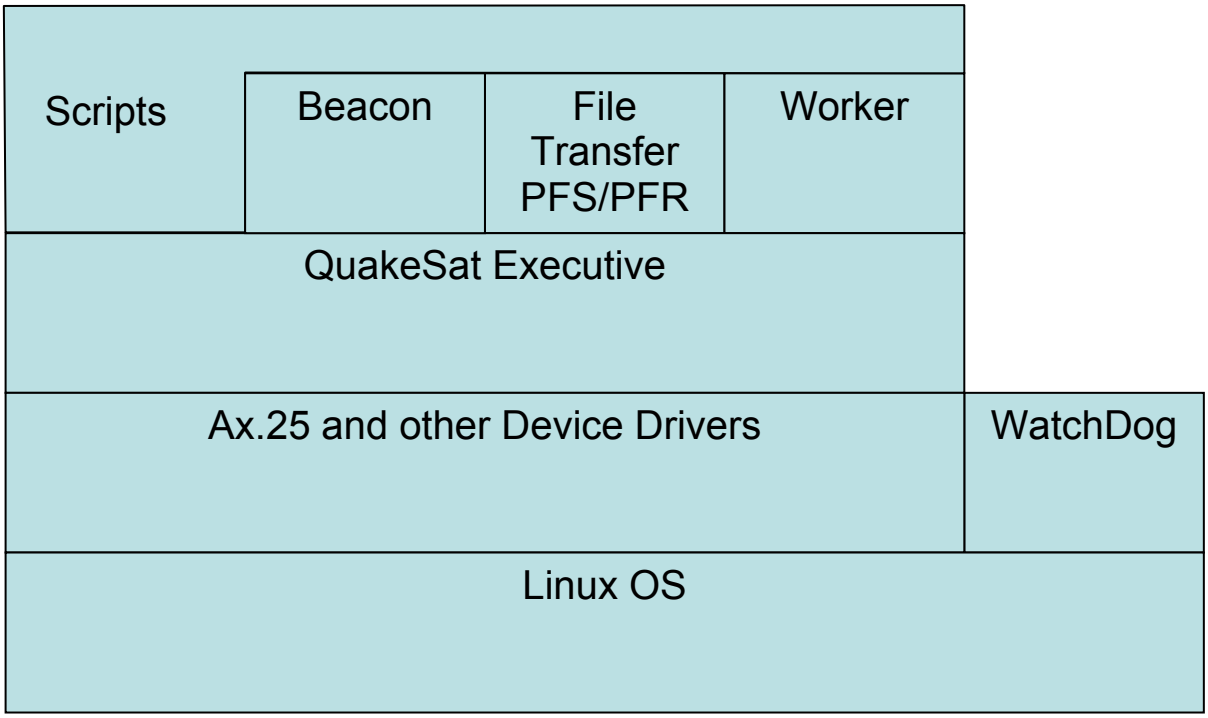
2.3.2.1 Satellite Software Architecture

The operational satellite software consists of a handful of independent programs. Each program is built as an executable that is generally independent of the other programs, or interfaces with a driver (one of the programs). This software architecture proved easy to develop and maintain while leveraging the power of the Linux operating system. Some of the programs were developed with minimal interaction between developers, reducing development cost, development time, and software complexity. When the software is broken into multiple programs, each of which can execute independently, and in one case,

designed so that multiple instances could execute simultaneously, the overall program risk is reduced. The risk that the software would lock up due to an untested race condition.

The simple software architecture can be outlined as follows:

- Device driver: hardware timers, payload board controller
- Executive program: receive a command, send a response (1 packet up/down)
- Worker program (reads in a time-tagged sequence of commands and executes them). This allowed multi-day planned collections.
- Beacon program (send beacons: alive health/status).
- File upload/download standalone programs PFS?PFR (with integrated hole management to fill in missed packets).
- Heartbeat generation program to tell the watchdog board all is working.



QuakeSat Software Architecture

2.3.2.2 Post-Launch Modifications

Despite the simple straightforward software design, post-launch testing showed that the software required modifications. The modular software design allowed software engineers to upload new code to the satellite without halting the system or performing risky replacement procedures. In most cases the replacement file was uploaded to a new filename. Ground software or satellite scripts were modified to use the new program name,

leaving the older program in place as a backup copy. In some cases, after the new code proved to be flight worthy, the old code was deleted or replaced with the new code.

2.3.2.3 Use of Linux Scripts

The satellite relied on operating system level scripts to perform maintenance tasks, control on-board software programs, and even execute some payload commands. The scripts, known as Linux shell scripts, were generally no more than 500 characters long. A new script generally meant changing one character (a character in a filename) and uploading a new version. Replacing and executing scripts under these conditions is very reliable.

2.3.2.4 Evolution to Automated Operations

Satellite operations evolved from a pre-launch best effort (make it work) mentality to an automated, highly-refined set of predetermined tasks. The satellite contact procedures improved over time by modifying existing scripts and adding new scripts to the satellite. A typical satellite script executed operating system level commands like "move", "copy" and "delete". The scripts used during a satellite contact often sent operational status down to the contact program. This immediate status capability was developed post-launch after it was determined that the status capability designed pre-launch was wasting time during a satellite contact.

2.3.3 Time

The accuracy of the timing of data is a central focus in many analysis efforts. Lack of good time tags can stop an analysis that would otherwise have been completed. Many hours of the analysis budget can be spent assaying the time once the mission has started, so it is definitely best to give it some forethought. Aptly time-stamping data is far more than a convenience, it saves on analysis dollars. Here are some key points:

1) What is the necessary accuracy of time required to successfully complete the mission? Is it necessary to know time to the accuracy of the sampling rate?

2) What is the precision (number of significant digits) of the time reported by the spacecraft. How does this compare to the accuracy requirement?

3) What is the basis of the clock upon which time on the spacecraft is indicated? Is it the CPU clock, subdivided by a counter? Is it an interrupt driven clock? Is it an external clock (often the computer is a watch battery clock) that is asynchronous to the processing operation?

4) How will clock drift be computed and applied?

5) What is the true sample rate, and how does it vary/drift with time?

6) What other sources of data must the spacecraft data be correlated against? How?

2.3.4 Data Collection and Digitizing

More flexibility in data collection would have been beneficial, for example by being able to create data collection files closer than 10 seconds apart, or by being able to collect longer files by using the entire memory buffer.

Since we were looking for ground-based 50 and 60 Hz power line harmonics with our magnetometer, it helps to use a sampling rate that is not an integer multiple of a common frequency (such as 50 Hz or 60 Hz power lines, 10 Hz, etc.) so that if such harmonics occur in the data, aliased harmonics do not sit directly on top of the non-aliased harmonics.

More digitizing flexibility would have eased collection of diagnostic data. For example, on QuakeSat it would have been useful to be able to digitize more than one channel at a time. However, this is often a more expensive or complex option, so it must be traded off accordingly.

2.3.5 File-based Download

Older spacecraft spew telemetry in a continuous stream of data which is stored in frames. For high-rate data, a part of the frame is often toggled to dwell on a particular measurand, inspiring the moniker "dwell" telemetry. Newer spacecraft, particularly microsats, often use computer files as the main currency for download. The operations game is thus much different than it has been in the past. Depending on needs, it may be necessary to track in real-time various files: on-board, partially downloaded, uncompressed but not verified, fully verified, etc. This can be tedious and expensive, and there is no substitute for planning in this area. If a LEO satellite lacks global ground station coverage, the operators must choose and schedule which files will be downloaded during a given spacecraft pass over a ground station. There is no substitute for a well-organized data system that can respond to shifting priorities during a fifteen minute spacecraft pass.

2.4 Mechanical Design and Construction

2.4.1 Cleanliness and Contamination Requirements

If the spacecraft has optical elements, there may be stringent cleanliness requirements. These may mean the build-up and test process must be done in a clean room while the staff is in "bunny suits". Even if the satellite has minimal cleanliness requirements, launching with other satellites on the same launch vehicle may require that your satellite be clean and not outgas in the vicinity of the other spacecrafts. It may require that the spacecraft undergo a "bake-out" process to force solvents to outgas before it is ready for launch. (This is another facility requirement for the right type of oven).

QuakeSat being loaded into its PPOD prior to shipping

2.4.2 Heat Dissipation

Heat dissipation can be a problem with certain components. In QuakeSat, the CPU, the power transistor in the transmitter, and the two DC-to-DC converters were relatively hot compared to the nearby satellite temperatures. A trick that we used was to place a special thermal conductive "rubber" like material (Gap Pad 1500 ™ from DigiKey), which is a type of pliable heat sink material that can conduct heat from a "hot spot" to a neighboring metal case or heat sink.

2.4.3 Solar Arrays (Mechanical Design)

The wing solar panels, although necessary for our power needs, were difficult to handle/work with in our setting. The thin sheet of aluminum that was used as the backing for each wing solar panel was not rigid enough to keep the panels from bending even under the slightest pressure. This resulted in several panels developing cracks during testing and P-Pod insertion handling. In addition, while in the P-Pod the wing panels were protected from the spacecraft body panels and the P-Pod walls by only two small plastic snubbers on each wing. The lateral vibration during vib testing also caused some damage.

The lack of replacement panels, repair time, and cost, required us to fly with these panels damaged. This could have been a flight-limiting problem had we suffered greater damage during the actual launch. We can only guess at the actual on-orbit damage to each array, since we lost the multiplexer which handled each of the panel currents. As it was, the satellite only started suffering power problems after the batteries failed in late January of

2004 and apparently only when it rotated to a poor power collecting attitude and it needed power for the payload and/or the radio.

A possible mechanical design change to help correct this problem might have been a hogged out area for the array in a thicker sheet (much like the body panels were mounted) or simpler, a hogged out "I" beam section that the panels sat in. This would have provided extra protection to the arrays and wires during the vibration of launch and would have eliminated the panel bending that occurred during handling and P-Pod prep.

The ridge could have also provided containment for the solar panel wires, the e-field antennas and the radio antennas to avoid catching on the P-Pod internals during deployment. This did not happen, but had one of the radio antennas caught on the corner runners of the P-Pod, the P-Pod spring would have been unable to push QuakeSat out. Another concern was the E-field antennas binding or catching on the corner runners or the door.

2.4.4 Machine Shop

Most mechanical parts need to be manufactured first as a prototype to check for fit and clearances. The team should either have a machinist in their group, or contract with a local machinist who can make parts on a quick turnaround basis, and if possible, should have previous experience making satellite parts.

2.4.5 Electrical Drawing Systems

Electrical circuit simulators (Spice and Electronics WorkBench) are good to pre-check circuit designs. PC layout programs (e.g. the free program Express PCB) are needed to define the circuit boards in a format that PC manufacturers can use for layout and fabrication. Four-layer boards are recommended so that ground plane layers can be added to minimize circuit noise.

2.4.6 Mechanical Computer-Aided Design (CAD)

The development of QuakeSat, from the initial design to the completed space unit, was greatly facilitated by using new CAD tools, which helped improve the overall efficiency of the mechanical design and development process. A CubeSat or microsat may be simple in comparison to larger satellites, but there are many mounting and fit checks that must be performed. A good 3-D drawing system like *Solid Works* can be very helpful in determining interferences before the parts are manufactured. Drawing systems such as *AutoCad* are also good for dimensioning the parts in preparation for machining and assembly.

This section highlights some of the valuable lessons learned in using *Solid Works*, the CAD program that was used throughout the QuakeSat project lifecycle.

2.4.6.1 Key Features of a CAD Program

The importance of using a CAD program was evident from the very start, when during trade-space analysis and part selection the foremost question that needed to be answered

was, "how would it all fit?" To answer this question, a powerful and intuitive CAD package where design ideas could be quickly incorporated and evaluated was used from the start.

After several QuakeSat parts drawings had come together, it became apparent that there were two critically important capabilities needed in order to perform accurate and efficient analysis ensuring that the designed parts equate to a system solution, and not just an individual part solution:

1) The ability to construct assembly drawings based on individually constructed parts and components.

2) The ability to impose position and motion constraints on parts in the assembled system, and to use the built-in automatic collision detection capability to locate possible spatial conflicts in the mechanical system design.

The ability to place constraints on the parts also enabled the dynamics of the system to be checked. *Solid Works* made both of these capabilities straightforward and easy to use and work with.

One example that illustrates the usefulness of assembly drawings with imposed position and motion constraints was in the redesign process of the P-Pod pusher foot. The redesign was necessary to accommodate the QuakeSat antenna mounts and the tip of the stowed sensor boom, both of which extended beyond the front faceplate of the QuakeSat bus. For both of these items, there were no viable satellite redesign options available, so we sought to modify the internal P-Pod structure to open the extra room needed to include these parts.

From a visual inspection, it was apparent that the original pusher plate could be modified to open up the needed space. Figure 1 shows the drawing of the original pusher footplate based on the dimensions taken from the preliminary P-Pod pusher footplate that was delivered to the QuakeSat team. Within a few hours time the drawing of the original plate was modified to open up access ports for the antenna and antenna mounts, and a hole centered about the middle of the plate to accommodate the telescoping sensor boom. Figure 2 shows the modified pusher plate that was used in the for-flight QuakeSat P-Pod.

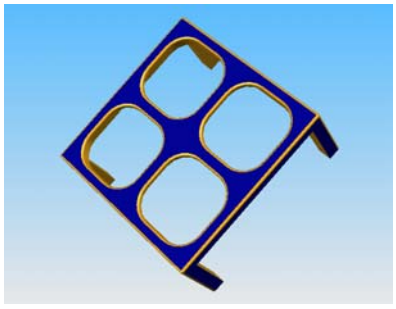


Figure 1. Original P-Pod pusher plate

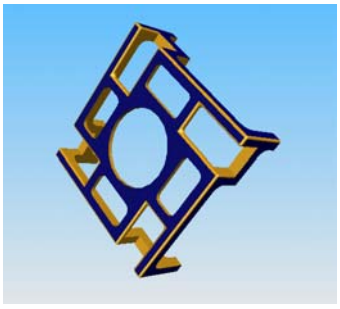


Figure 2. Modified P-Pod pusher plate

The redesigned part, in this case the pusher plate, by itself is of little value if it does not satisfy the needs of the entire system, which here is to adequately couple to the top faceplate

of the QuakeSat structure while giving critical parts ample room and clearance so that there is no potential for them to be damaged during storage, launch, and ejection/deployment.

As a result it became valuable to use the ability of imposing motion and position constraints to model the degrees of freedom that the actual pusher plate has in its motion along the axial length of the P-Pod as the P-Pod master spring uncoils and ejects the satellite. To do this, the Integrated QuakeSat model (with antennas and folded solar panels), pusher plate, compressed P-Pod spring, and P-Pod assembly were all combined into one system assembly file including constraints that were adequately placed to model that of the actual system. Automatic collision detection was then used to identify any mechanical design defects or oversights. This uncovered several conflicts that were easily resolved with just a few pusher plate modifications. Then, using the final cleaned up pusher plate part, no spatial conflicts were reported throughout the range of motion.

The partial integration of the P-Pod pusher foot, spring, and QuakeSat top plate assembly, as well as the fully integrated P-Pod and stowed QuakeSat assemblies are shown in figures 3 and 4.

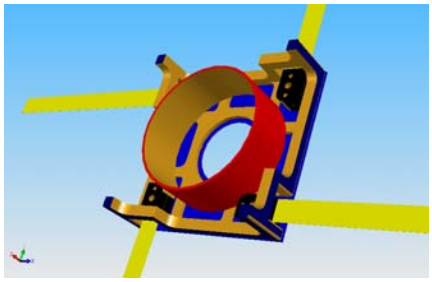


Figure 3. Pusher Plate, Spring & QuakeSat Antenna Assembly

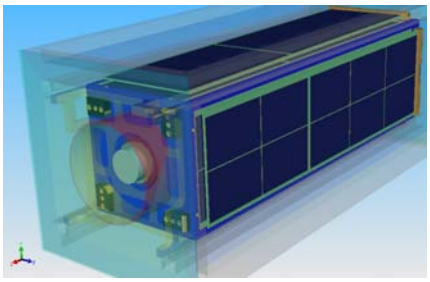


Figure 4. Integrated Stowed QuakeSat & P-Pod Assembly with Imposed Constraints

At this point the pusher plate part was auto converted to a drawing file, which was then subsequently handed off to the machinist for fabrication. The modified P-Pod pusher plate was so well designed that the first machined part was the only one needed to be made. There was no need for process reiteration, saving us both time and money.

2.4.6.2 CAD is not a Complete Answer

Another lessoned learned through this very same process of P-Pod pusher foot redesign was that a good CAD program couldn't solve every problem. Dialogue and good communication with the machinist is crucial. As it was for the modified pusher foot, the entire design looked good both on the computer screen, and on paper, but there are still some details that cannot easily be realized in a metal structure. So some further minor changes were necessary at the time of machining that were then flowed back into drawings for accuracy of record.

2.4.6.3 *Handling Multiple Assembly Configurations*

An additional valuable feature in a CAD tool was the ability to store and save specific configurations of an assembly file, since it is helpful to switch between modes of operation to conduct further mechanical design analysis. The following two images are of the integrated QuakeSat small satellite, both of the same file, just in two different configurations.

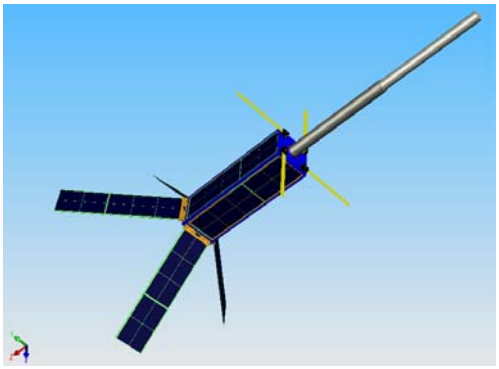


Figure 5. Integrated QuakeSat Assembly, Deployed Configuration

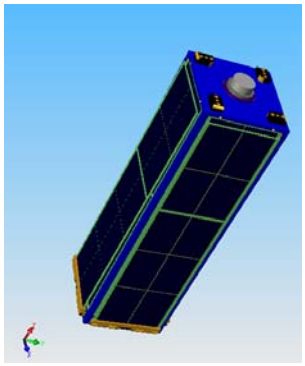


Figure 6. Integrated QuakeSat Assembly, Stowed Configuration

2.4.6.4 *Use of Free Viewer Programs*

The ability to use freely downloadable viewer applications greatly enhanced the communication of design concepts between team members, the customer, and the project machinist, as well as to the general public by being able to generate Web browser enabled viewable drawing files.

2.4.6.5 *Additional Mechanical CAD Suggestions*

For brevity, the following are additional suggestions that may help future small satellite teams to better utilize today's mechanical CAD tools for efficient design and analysis.

- Take advantage of assigning material properties of parts so that assembly drawings auto-compute the mass and moment of inertia of the modeled system
- Take advantage of finite element and stress analysis tools that are now being incorporated into many standard CAD suites, which supports reuse of the CAD models
- Take advantage of photo-realistic animation tools that are also being incorporated into some CAD suites for enhanced communication through visualization, which can be used and shared on the Web, in presentations, and in project related media
- Take advantage of commercial component and part interpreters that are available with some CAD programs. (For example, Solid Works has an advanced module called Component Works which can be used to auto generate accurate component models such as connectors in several seconds for use in CAD drawings based on basic geometry data

and images, which many companies are now providing as supplemental material to their online part description, documentation and specifications.)

- Take advantage of drawing revision control systems that are now being incorporated into many CAD programs
- Take advantage of to-scale PDF format drawing files for electronic transfer and archiving of drawing files, when this capability is incorporated into CAD suites.

2.5 Power System

QuakeSat had a number of challenges with the power system.

2.5.1 Design Margin

The power system design should have adequate margin for all the loads in the satellite, and should de-rate everything (to 50-70% of spec).

2.5.2 Telemetry Sensors for the Power System

Individual current monitors (Maxim 471 chips) are recommended on the solar panels so that you can determine which panels are working well.

Current monitors should be used at the input to the batteries and the (positive only) DC-to-DC power supplies. Load current monitors are also recommended for the payload subsystems if there are switches available to turn these subsystems on and off.

Voltage sense points should be included at the solar bus, the batteries, and the load bus (output of the DC-to-DC power supplies). One should be careful to include resistor divider networks so that the telemetry voltage point (input to the A/D) never sees voltages higher than the range of the A/D (e.g. 5V DC). The power supply voltage should have very little ripple, however, and may require capacitors and chokes to reduce the ripple.

Temperature points can use the LM-34 (deg F) integrated circuit to read deg F or C, and if a ballast resister is used with a negative power supply, the sensor will have a range down to minus 55 deg. F.

2.5.3 Solar Panels

The solar cells typically have "bypass diodes" which help if an individual cell "opens". The panel assemblies must also have "blocking diodes" to prevent reverse current from flowing through those panels that are in the "shade" when other panels are in the sun. These blocking diodes are separate from the panels and must be added in the overall circuit.

The spring-wing concept was necessary to support our power needs. It could be used to provide even more power with additional wing panels depending on the mission.

2.5.4 Batteries

There should always be redundant batteries. Dual batteries reduced our single fault failure risk with respect to power. However, QuakeSat's batteries did fail within 4 days of each

other, likely due to similar operating environment (over 110 degrees F and possible bake out of the electrolyte).

The 2 Li-Ion batteries used in QuakeSat are very sensitive to any deep discharge events. For this reason, the batteries should have a flight disconnect switch so there is no way they can be discharged (and destroyed) during testing. An external connector, available through the P-POD, is necessary. This connector must be able to bypass the flight batteries for testing (all test power should be supplied using ground support equipment). A single GSE power box was built with flight-like batteries, Li-Ion charge controllers, and voltage and current meters to monitor the system. Fuses in the GSE are a good idea in case an accidental short circuit occurs in the flight equipment.

The Li-Ion batteries should be sealed and mounted in a pliable mounting compound (e.g. RTV) since the manufacturer claimed that the batteries would slightly "expand" when charged, and contract when discharged. Our experience was that the batteries (unsealed) lasted 7 months in a sun synch orbit (temps varied from 90-110 deg F in the early rev eclipses, to 110-120 deg F in the constant sun periods). They may have "baked out" their electrolyte, since they were not sealed beyond the normal factory packaging.

2.5.5 Charge Controllers

In the case of Li-Ion batteries, there should be separate charge controller boards for each battery.

The charge controller circuit (pulse-width modulated charging circuit) may generate unwanted EMI, and should be tested in the final configuration to see if they generate any noise on the power lines that could affect the instruments on board.

2.5.6 Power Supplies

The internal power supplies and analog boards should have large tantalum capacitors and ferrite beads to remove ripple. The DC-DC power supplies (80% efficient) are not very quiet (1-5% ripple). They also dissipate a good amount of heat (100-150 deg F in the sun synch orbit with constant sun on the satellite). If low power supply noise (ripple) is a strong requirement, consider using voltage regulators (only 40-50% efficient depending on the difference between the input battery voltage and output voltage to the load). They also generate more heat and must have very good heat sinks (conduction paths) to a cooler part of the structure.

2.6 Payload

2.6.1 Payload and Downlink Interference

For our payload, our communications downlink interferes strongly with our magnetometer sensor. One way we mitigated this was to add a second ground station at a distant site, so we could collect near one ground station and download using the other one. There may be

other ways to mitigate this that involve filters that isolate the low frequency sensor from the high frequency radio.

2.7 Other Sensors

2.7.1 Telemetry Multiplexers

The QuakeSat telemetry system had over 30 analog data points, and two analog multiplexers (16 inputs to 1 output) were used with the 16 channel A/D converter on the Prometheus CPU card. It is important to keep all the analog inputs below the supply voltage on these multiplexers. A solid ground (ground plane layer in the PC board) is also a good practice.

The very early failure of one of our two telemetry multiplexers eliminated roughly half of our housekeeping sensors, primarily solar panel currents and other internal currents. An interleaved design (some temperatures, currents. voltages on one multiplexer and the rest on the other) would have reduced the impact of losing a multiplexer. (During the mission we would have traded a few subsystem temperatures for even one or two solar panel currents.) However, an interleaved design would have been more complex and could have caused other issues, i.e. it would have been a little more difficult to layout and integrate (software, telemetry, etc). The spacecraft overall had a high risk to single fault failures. This was only one such risk and the cost to catch and correct them all was not within the project budget or timeline.

2.8 Reset Mechanisms

It is important to have an alterative way of resetting the communications system or rebooting the spacecraft. The CPU will probably have one or more "single event upsets (SEU)" during its lifetime on orbit. This means that a high energy particle has hit a circuit (usually RAM) and the computer halts. It is mandatory that some type of watchdog circuit monitor the CPU activity, and to "reset" the CPU if it halts.

QuakeSat had a watchdog circuit board (built by the team) that would automatically reboot the CPU if the heartbeat was lost for 90 seconds. A PIC processor monitored the "heartbeat" signal from a serial port of the CPU. If the PIC program did not detect a transition of the heartbeat signal within 90 sec, it sent a "soft" reboot signal to the reset pin of the CPU. If this failed 3 times, then a "hard reset" was sent by toggling the CPU power switch (2 power transistors on the power board) so that the CPU went through a complete cold reboot sequence.

QuakeSat also had a DTMF (dual-tone multi-frequency) detector chip that listened for a certain sequence of touch-tone-phone tones. The DTMF chip was connected to the radio receiver and, if it received a special sequence of 4 tones, then it reset the CPU (a soft reset 3 times, then a hard reset if those failed).

We also used another method, namely the "reboot" function in Linux to reset the computer if any process was corrupted.

All these resets were used on orbit at various times.

3 Ground Station

3.1 Early Ground Station Activation

It is important to activate and test the ground station prior to launch. This task is easy to overlook, since shipping the spacecraft often becomes a critical focus of resources. Ideally, the ground station would be used to checkout the spacecraft while operators learn to use it. Tests would be run using actual command and telemetry methods to acquire data, and the entire system would be wrung out in advance. This implies of course that the ground station be operational before the integration tests begin. If it isn't, then the spacecraft team will develop other means of getting their jobs done, often to the detriment of the program. They might, for example, take data in a form that will be never become a part of the spacecraft database. Or they might shun the ground station data processing as an added burden to their task. Thus the ideal is that the ground station be activated, validated, and familiar to everyone before even the early spacecraft testing starts.

Ground Station Equipment Rack at Fairbanks

3.2 Ground Station Software

The QuakeSat ground station software performs three primary functions. The software controls the radio, it controls the antenna, and it allows a remote user to schedule and manage satellite contacts via an integrated Website. Underlying these three functions is a database that stores satellite and scheduling information, a program called "predict" that tracks satellite location, various log files that store ground station component states, and software that manages the transmission of data between a remote contact computer and a satellite. This makes the ground station an intelligent conduit between the satellite operators and the satellite, but it does not create or interpret any of the commands between the contact computer and the satellite.

All of the ground station functionality can be set and viewed through a web site that runs on the ground station computer. This allows the ground station to be deployed at any site that is Internet-enabled. Besides the Web interface, an Internet camera was added to the

ground stations and proved to be a beneficial debugging tool. The camera, focused on the radio and modem, validated when the contact computer was trying to talk to the satellite, when the satellite was transmitting, the transmission/receive power levels, and if messages were received but not decoded completely. Visual feedback over the Internet from a remote ground station allowed Quakefinder personnel to monitor the quality of satellite contacts from any Internet-capable computer.

3.2.1 Architecture

The architecture uses a flexible layered approach to communicating with a satellite. Since the satellite command and control is not intertwined with the ground station functionality, debugging and deploying the ground station is very straightforward.

The ground station software architecture takes advantage of virtual machines (VMs), which facilitated:

- Recovery. We used VMs to copy primary ground station computing resources. These copies could then be run on backup CPU hardware with minimal effort.
- Multi-mission support. We used VMs to provide local customization of ground station resources by the satellite operations team. This is a simple mechanism to implement that enables great functionality.

It is important to avoid losing bits once they get to the ground station, since it is expensive to get bits from the satellite to the ground station.

The ground station needs the ability to run application-level code at the ground station. This facilitates maintenance and post-launch flexibility.

3.2.2 Verifying Antenna Pointing

A second camera focused on the ground station antenna would have been an additional debugging aid. Proper antenna pointing is a crucial aspect to ground station operation and if site personal are not available to assist with antenna sighting, a camera can be beneficial. A camera may also give a clue about weather conditions like excess ice on the antenna although we have not identified an all-weather camera. One problem encountered by the Quakefinder team was a bug in the antenna control code. While the antenna was sighted correctly, the software was directing it to the wrong azimuth angle. Visual verification would have eased detection of this problem. While the problem was solved with human visual verification, a proper inspection of the antenna log files also verified this problem. This brings home the point that robust operational logging is extremely worthwhile in this world of 40-gigabyte hard drives.

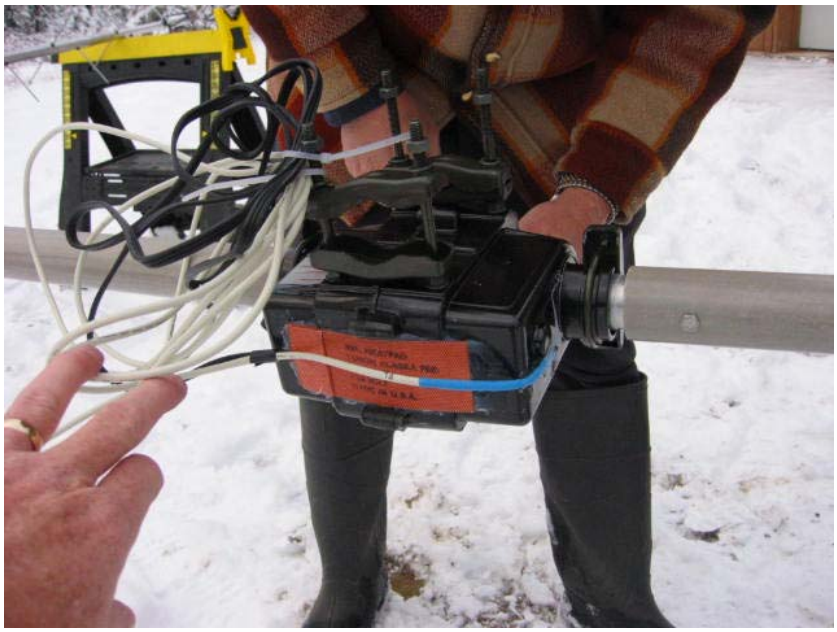
3.2.3 Second Ground Station

Quakefinder deployed a second ground station 30 miles east of Fairbanks, Alaska. The station frequently operated at temperatures below –10F successfully with the aid of 100 watt

patch heaters on the antenna azimuth- and elevation-drive gears. The heaters protect the gears from freezing.

The Fairbanks ground station is near the North Pole. Before the ground station was deployed, it was determined that most of the satellite contacts traversed a course that would require a keyhole operation by the antenna (a 360 degree azimuth slew). A modification to the antenna driver software and the physical direction of the starting position of the antenna eliminated almost all of the keyhole conditions. While this change initially contributed to antenna pointing problems (due to errors in the antenna driver software and the difficulty of remotely debugging it), all of the issues were resolved, further enabling efficient contact time operations.

A benefit provided by the Fairbanks ground station was the increase in daily satellite contacts. The satellite followed a polar orbit so the addition of the Fairbanks site increased the number of contacts from six in Palo Alto to seventeen at the two sites. Since the ground station Website required each contact to be scheduled manually and individually, an automatic scheduling capability was added to the Website. The exact implementation can vary, and Quakefinder opted for a schedule file that required uploading to the ground station. A simple click of a Web page schedules the next forty contacts. Another click deletes traces of expired contacts. This capability saves enormous time over the course of a long mission.



The Fairbanks elevation drive motor with heater being installed

At the end of the QuakeSat mission, this new ground station will be given to Stanford University for use with future UHF Cubesat and microsats. This high latitude station will be of great use in supporting the polar and other high inclination orbits these satellites are primarily being placed in.

3.2.4 Maintenance

The ground station software is built on the Linux operating system. This means that remote security, debugging, and maintenance capabilities are built into the operating system. Once deployed, the Fairbanks ground station software was modified remotely from Palo Alto. Often bugs could be detected, debugged, and fixed between satellite passes.

We had some issues with the Internet connection to our Fairbanks ground station, mostly due to the remoteness of the site and bad winter weather that hindered repairs.

3.3 Ground Station Telemetry

There is never enough telemetry. Telemetry is hindered since some COTS solutions are black boxes with less telemetry access (e.g. radios). Telemetry also requires more resources, such as CPU cycles and network bandwidth.

Remote ground station operation was facilitated by increased telemetry (i.e. the remote camera).

There is a definite need for local ground station autonomy to process telemetry and recover from errors automatically.

Telemetry is occasionally needed for manual recovery of ground station components.

3.4 Ground Station Testing

Complexity of the system made distributed debugging difficult. The engineering model of QuakeSat enabled software and hardware debugging post-launch. This facilitated on-line testing of the ground station, full end-to-end testing, debugging, and recovery.

4 Ground Control Software

The QuakeSat contact software is a Linux graphical user interface (GUI) application that communicates via an Internet connection with the two ground stations (described above). The software transmits commands to the satellite, receives responses, and can transmit and receive files. The software also maintains satellite and ground data file status. During normal operation files are created on the satellite. They must be downloaded to the ground during a normal satellite contact. Sometimes the entire file is not downloaded in one pass, so state must be maintained until the complete file is downloaded and made available to analysis. This means that the contact software must remember file state from previous contacts.

The contact software must follow a design criterion that revolves around one central point. If contact time is limited, the contact software must be efficient to use. This means very little typing and the use of commands that perform as many operations as possible. GUI buttons that represent multiple commands to perform specific operations saves time. Programmatic generation of filename lists to download means typing errors are eliminated.

4.1 Automation

Automation of contact software is key to reduce errors, maximize contact time, and minimize wear and tear on the support staff. The QuakeSat contact software evolved from a GUI program that required the operator to know a half dozen operations and recognize error conditions based on specific satellite responses to a fully automated program that required no human intervention. This meant that Quakefinder could support a 24/7 operation with minimal labor cost. While not every operation of the satellite was automated, two primary tasks, file download and flash memory management, were automated. This left command file upload and extraordinary maintenance operations as manual tasks. The contact scheduling of the ground at Fairbanks was also automated, allowing 2 months worth of contacts to be scheduled at a time with minimal cost.

4.2 User Interface

Automated contact software requires a number of support files to guide the program through days or weeks of operation. This includes contact schedules, file lists, and various files that are utilized in normal maintenance. These files can be quite cumbersome to maintain with standard editors. Quakefinder developed a Website that allowed an operator to control the automatic operation of the contact software without knowing how to use a Linux workstation or Linux tools. This made day-to-day operations simple and generally error free.

4.3 Status, Log Files, and Error Handling

The Web site also contained all of the log files generated by each contact. Every contact software operation was time-stamped, recorded, and saved. The log files proved invaluable during the early days of the mission. The logs were used to determine round trip transmission delays, human operator error, and satellite health. Over time important information within the main logs were identified and used to generate specific logs that could be easily distributed to members of the satellite support staff. A summary log of important contact information was generated after each satellite contact.

The contact software used e-mail to distribute logs and warn satellite operators of health problems that were detected by the contact software. Health issues included satellite reboot, satellite malfunction, and summaries of memory and disk utilization. The QuakeSat flight operations included sixteen contacts a day seven days a week, so automating error reporting alleviated the support staff from reviewing contact logs, thus reducing operational cost.

4.4 Improving Satellite Communications

Operationally, one issue that proved difficult was when we had poor communication with the satellite, either because of poor satellite antenna orientation, noisy background space environment over the pole, bad pre-amp early in the mission or low satellite transmitter power late in the mission. Initially it was cumbersome to determine if commands executed. Either the response was lost or the response only indicated that the command had reached the satellite and executed. The execution status required further commanding. Over the

course of the project the QuakeSat team developed satellite scripts that executed commands and returned results by transmitting the return status three times. The contact software filtered out the redundant answers and presented the results once. This approach resulted in a more efficient use of satellite contact time. The chances of an immediate and meaningful result increased significantly requiring less operator time.

The issue of lost responses must be designed into any system that will have communication issues. Even in a good communication environment there will be events that result in dropped responses. This is true in both directions. QuakeSat avoided any command response situation where the satellite would hang if it did not receive the proper response. On the other hand it is challenging to design a command response architecture where all responses could be lost. One solution is to design commands that can be issued to the satellite multiple times in a row without causing harm. An alternative is to simply design a mechanism that allows the operator to request the previous status and be certain that the response belonged to the correct command.

4.5 Post-Launch Satellite Command Changes

The flexibility of the satellite architecture allowed new commands to be added post-launch. Maintenance commands were neglected prior to launch and developed while the satellite sat on the pad and post launch. The contact software was quickly changed to adapt to the new satellite commands (scripts).

5 Subsystem/System Integration & Test

5.1 Third Party Hardware

A number of hardware items used on QuakeSat were purchased off the shelf for use during the mission. In general this worked out well. But there were some differences in how things unfolded. The first is that detailed use of anyone else's hardware means spending time seeking more detailed answers in the user manual, obtaining circuit diagrams, etc. Here, glaring differences showed up between vendors. Some had answers, others did not. Our solar cell vendor for example, set up a teleconference with their resident PhD, and our questions were addressed. Some manufacturers were less supportive in answering repeated data requests about their products. In fairness, we bought only a few products from them. The enticement of having your company name associated with a spacecraft has lost some of its novelty, apparently. For the spacecraft components then, availability of good documentation, support (perhaps a separate support contract might work well), and user-group advice can become crucial.

A second issue cropped up when we found that several of our hardware items, like the batteries, had added electronic components in them such as fuses, etc. that were not mentioned anywhere in the literature, or known by their reps. These items turned out to be crucial to mission performance, but it was months into the mission when they were discovered.

A third issue concerned attempts to modify/troubleshoot a third party's circuitry. In one situation, when we compared the circuit boards of two of our flight radios, both of which had the same part number, we found them in fact to be different. One of the radios had blue-wires and different circuitry. The two gave us fits trying to assess why they behaved differently until Jamie Cutler noticed the extra wiring.

5.2 Computer Integration and Testing

Even with simple computers you will probably go through 1-3 upgrades of the BIOS. Two problems in particular caused several weeks of lost time.

Build out your development environment. One should have 3 engineering units in parallel for development and testing. We needed to take more time up front.

We needed better tools. We needed engineering models of the power and payload boards sooner, again with multiple copies. We needed more scope work.

We should have left the Ethernet interface on until the end. The low-bandwidth serial interface caused lots of frustration and delays.

5.3 Power System Testing

5.3.1 Solar Panel Testing

The solar cells should be individually tested before assembly into solar panels. They should be (and were) "flash tested" to measure their output (voltage-current profiles). They should be retested after each major test (system functional test, vibration test, thermal-vac test, etc). Often a micro-sat developer must use a third party resource to accomplish flash testing, and often this is done prior to final integration of the spacecraft. Taking a lesson from a larger aerospace firm, we find that they flash test with the goal of quality control, in addition to the goal of system characterization.

Solar cells are fragile, and should not be assumed intact at the time of delivery. There should also be accessible test points (electrical connections) which can verify array operation after the spacecraft has been fully assembled. An in-house, simple, and repeatable solar cell lighting test is essential for verifying that the array circuitry is intact after final assembly, troubleshooting, launch pad transport, etc. Our final system test was done in the local parking lot using the actual sun since we did not have access to a "sun simulator" at that point in the testing. You do what you must do to get the job done on time.

5.3.2 Use of a GSE Power Supply

Often, ground support equipment (GSE) power supply is used in lieu of the spacecraft's batteries and solar panels for testing and integration. Experience has shown that the GSE power supply can obscure the true system noise that is visible when only the solar arrays and batteries are involved. The differences can be even more pronounced when the spacecraft's charge/discharge cycle starts. In the case of QuakeSat, the mission had a very

sensitive magnetometer, and the Li-Ion battery chargers generated noise that was not detected until the satellite was on orbit.

5.3.3 Solar Array Rotisserie Testing

For a micro-satellite, there might be insufficient funds for a solar simulator to put the spacecraft in simulated full sun for the length of time required to characterize system noise, much less a full charge/discharge cycle of the batteries. Such testing limitations leave a number of risks unresolved, most of which are obvious.

QuakeSat and team performing the solar array rotisserie testing in the office parking lot.

Outdoor sun can be used to reduce risk, particularly in dry climates like California, but then meticulous planning will be required to actually pull off the test. First, the spacecraft will rotate in a bath of sunlight which strikes from all sides, including underneath. A test jig must be built which can rotisserie the spacecraft at a nominal rate (if the arrays are not too fragile in their deployed state). Transporting the spacecraft from the assembly area to the

open sun, deploying arrays in a 1-g environment, rotating the spacecraft while simultaneously collecting data (either via radio link or radio by-pass umbilical cable), keeping cables, pencils, etc. away from the solar arrays, is difficult and nerve-wracking. Compounding this is that often such a test cannot be done until very late in the delivery schedule when the spacecraft has been mostly assembled. Reduce these concerns by planning these tests early and well.

5.4 Communications Subsystem Integration and Testing

Radio testing and communications issues can be tricky, especially given that RF waves can behave significantly differently on the ground due to the nearby ground plane and line-of-sight effects. A very effective choice is to place a test connector on the outside of the spacecraft that can always be used to bypass the radio for purposes of command/telemetry and other subsystem testing. An ideal design would make this service available at any time during integration, thus allowing radio and communications issues to be isolated from the rest of spacecraft integration.

Communications is nominally presumed to be "good" and thus will not be highly visible during mission operations. Striving to make the ground station's interface to the spacecraft identical for both radio and umbilical modes will gain the mission team significantly more experience in operating the spacecraft, and allow the software delivery task for the other subsystem not to be impacted by the communications subsystem's integration.

5.5 Sensor and Payload Testing

Our low frequency magnetometer is very sensitive to magnetic and vibration noise, making it difficult to test, except perhaps with a large, expensive magnetic isolation chamber. The environmental noise of a typical office/industrial area can swamp the instrument's response and make pre-flight characterization problematic. Sampled signals in particular, created by an analog-to-digital converter, can be seriously affected by tones and other internal noise.

Incorporate creative ways to test it, for example by disconnecting the sensor. Test the hardware by: (a) looking for artificial signals leaking into the sensor circuitry or digitizer when the sensor is disconnected; and (b) inject signals into the system, with and without the sensor connected.

Use of a dummy source across the input can give a much better picture of internal system noise. In a system which has selectable modes or channels, it is an easy matter to include one setting which places a resistor across the normal sensor inputs. Sampling of the dummy-loaded system can greatly simplify the job of separating internal noise from the real signal, especially when on orbit.

5.5.1 Calibration

Each case is different, but care should be taken to have accurate, calibrated "signals" and all exterior "noise" should be minimized.

5.5.1.1 Magnetometer Calibration

Calibration of the instrument, in this case a sensitive magnetometer, requires a triple gauss chamber. We used one at UCLA, but the test set up was difficult because the magnetometer had to be separated from the satellite, and the connecting cabling and satellite were subjected to the noisy lab environment.

5.5.1.2 Filter Calibration

Before shipping, measure the frequency and phase response of every filter well into the stop bands on either side. If the filter is an anti-aliasing filter, measure it from 0 Hz to well beyond Fs/2 (where Fs is your intended sampling frequency), so that the effects of aliasing can be understood. The measurements must be capable of supporting post-collection whitening (flattening) of the collected data spectrum. In addition, digitize the output using the expected sampling rate and measure the frequency and phase response over the zero to Fs/2 interval. Synthesize signals beyond Fs/2 to help characterize the effects of aliasing on the digitized signal.

5.5.1.3 Optical Sensor Field of View (FOV) Calibration

Optical system calibration also requires special chambers and scene simulators.

All optical sensors, including diode IR sensors, should be tested for Field Of View (FOV). There are a number of conditions that these sensors will see: direct sun straight into the FOV, dark space in FOV with sun scattering into the FOV, albedo pointing, moon pointing, deep space pointing, etc. Without test data and realistic approximations to the expected signal levels on orbit, the signal levels and possible outcomes are simply too large and uncertain to infer. If IR sensors are used for crude attitude determination, include light baffles for narrow field of view, and include 3 orthogonal sensors to get unambiguous readings.

5.6 P-Pod Testing

A planned firing and opening of the P-Pod failed when we were unable to charge the circuit before the test. This reduced the P-Pod deployment testing to once, and that was before a major P-Pod repair was done, i.e. the flight circuits were never tested by QuakeFinder. (Some pre-ship testing was done at Cal Poly.) The P-Pod release mechanism has undergone a major redesign, and is much improved. Testing the release mechanism with a flight-like satellite is still essential.

5.7 System Testing

System testing should be well planned with a testing hierarchy. Given the limited time available for ground station development, limited full-up testing was done with the ground station. A complete end-to-end test with the satellite, the RF link, and the actual ground station is highly recommended. Transporting the satellite to the ground station area (e.g. Stanford if that station is used) is recommended.

5.7.1 Test Points

Additional test points are always needed on the system. We could have designed more test points into the satellite for testing, and added software for making additional measurements. In the case of the P-Pod, only one external connector was allowed on the satellite to comply with P-Pod rules.

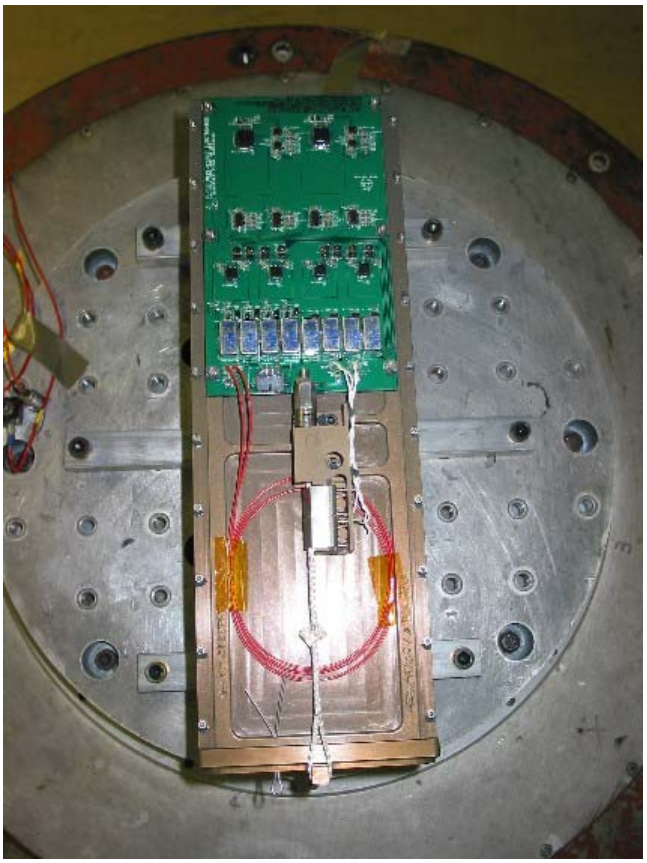
5.7.2 Bench-Top Satellite Testing

We did a lot of bench top testing, but you can almost always do more. Saving the test data and recording the activities done in the test are important. A time annotated description of the test will be required if you ever want to look at or evaluate the data again later; one's memory will not be good enough.

5.7.3 Mechanical Testing

5.7.3.1 Vibration Testing

Vibration testing helps ensure the satellite can survive the trip to orbit and can operate, and to ensure the other payloads on the launcher are not impacted by a failure on your satellite. Although both are important, the second is paramount if small satellite missions are going to remain viable. Loss of a booster or primary mission due to the failure of a secondary payload is the biggest reason it is difficult to get on a booster as a secondary payload. QuakeSat has accelerometers on the P-Pod, but not on the interior of the satellite (no room). If you have room to place a temporary accelerometer on a sensitive element of the cubesat, and to bring the data out through the umbilical, do so.



QuakeSat in its P-Pod on a vertical shaker table

5.7.3.2 Thermal Vacuum Testing

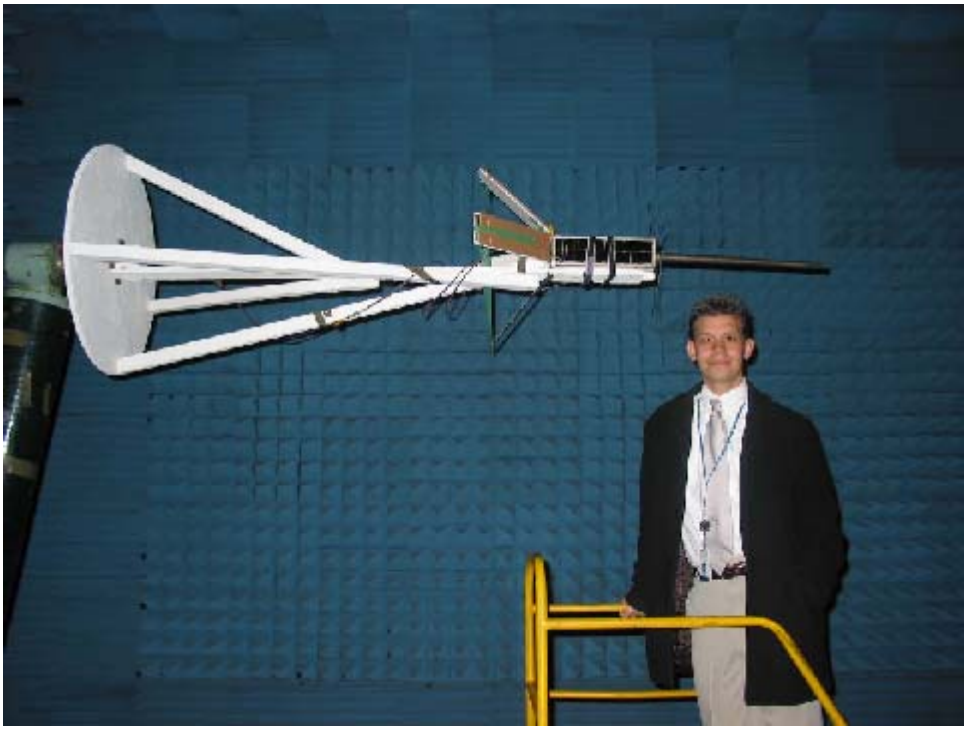
Overall the testing went OK. The issue of corona discharge (i.e. the distance a spark/short will jump is dependent on the air density) came up just before we decompressed the thermal-vac chamber. We had planned to leave the spacecraft power on during the decompression so we could test the flight batteries, but changed our test scenario since we were going to be un-powered during the low air pressure "corona" period of the mission as well. However, it caught us by surprise at the time of the test.

Because of the foregoing, you will need to turn the spacecraft on and off while in the vacuum chamber. Be able to bypass the flight disconnect switches, and provide access to the power system while the spacecraft is in the chamber.

5.7.4 Test Mounts

We almost lost the satellite here. Anytime the spacecraft is going to be suspended in other than its typical work bench fashion, exercise caution. Mountings during most tests are going to be unique—make sure they are secure and sufficient to hold. The spacecraft typically has solar panels on most outside surfaces, so it may be difficult to find a mounting

or clamping surface. During our RF anechoic chamber testing only days before shipping, the spacecraft was fastened to the test fixture with several bands of Velcro. During rotation of the satellite in the anechoic chamber, it slipped in the fixture, but did not fall. Had it fallen, the spacecraft would have likely been destroyed or at least un-repairable before our ship date. A few months later, near where we had done our own thermal-vac testing, the NOAA-N Prime spacecraft fell off its mount just prior to testing because it was not secured. It was a multi-million dollar accident.



QuakeSat in the Lockheed-Martin Antenna Test Chamber

5.7.5 System Testing Schedule

Our system integration and test period was highly compressed. This led us to drop many tests that normally would have been performed with more time. The time required for this phase should not be underestimated, especially in a more loosely defined student type project. (Things will not work/integrate exactly as expected or planned.)

The lack of time for these tests lead QuakeSat's primary payload to be less sensitive than we had hoped: more internally generated noise from other subsystems, noise on the lines to the A/D converter, variable (because of ripple) ground floor measurements.

When the time truly comes for integration and system level testing, flexibility will be required. This is even truer for a secondary payload since the launch timeline will be fixed and controlled from outside your team. The spacecraft must have some minimum power and communications operating capability when it is shipped to the launch pad, otherwise

you may never hear from it again. In most cases, some data will be more valuable than no data.

5.7.6 Flight-Like Full Satellite Testing

At least some satellite testing should (must) be full up and flight-like, i.e. on internal batteries and solar arrays, with beacons and devices on, telemetry and payload data collections, etc. You won't be able to recreate flight, but you should have at least one test that has the above mentioned items working all at the same time (if that is flight-like).

The first and only time we had sun on our solar array was 2 days prior to shipping; even then it was a short, limited test. We didn't have sufficient time, to run any payload collections in this mode, ie the charge controller handling power from the solar arrays. This would later prove to be a source of addition payload noise. In addition, no testing was done with radio communications or beacons running during a magnetometer collection, as originally intended during the mission. During the mission, we worked around this by turning the beacon off and not planning any radio communications during magnetometer collections, but this problem was not discovered until after launch.

5.8 On-Orbit Integration and Test

Much of our additional integration was completed while on orbit. This is not recommended, but the compressed timeline forced this and removed the option to change/correct/calibrate many items on the spacecraft.

Many software and operational improvements were successfully integrated into the system after the spacecraft was on-orbit. This was made possible by the modular nature of our command and control system. This allowed us to have a simpler pre-launch concept that was refined when the satellite got on-orbit. That is, we spent little or no time and effort on items that failed early in the mission or became too difficult to achieve within our budget and time constraints.

6 Flight Operations

6.1 Spacecraft Mission Phases

In general all spacecraft missions can be broken into several phases. The launch and initialization phase, the spacecraft characterization phase, the mission phase. Each of these general phases can be broken into sub-phases. The spacecraft characterization phase can be broken into early characterization, trending characterization and end of life (EOF) characterization sub-phases and in some form extends thru the entire life of the spacecraft. The mission phase can be broken into early mission, nominal mission and end of mission sub-phases and starts after initialization and early characterization are complete and ends when the spacecraft can no longer support the collection of mission data. Note: the primary mission may have ended long before the end of the mission phase is truly over; secondary missions or further characterization may be also done during this end of mission phase.

6.1.1 Initialization Phase

Problems during our initialization phase were primarily due to the loss of multiplexer-2 at launch or shortly afterwards, a bad ground station pre-amp (pre launch testing had shown this to be suspect, but we didn't have sufficient time to change it out and test before launch) and uncertainty in exactly which space object we were for tracking (there were 8 objects ejected by the launch vehicle). The bad pre-amp was the biggest problem due to its direct impact to communications

Limited pre-ship radio communications thru our ground station made many of our first contacts learning experiences as we developed the technique to reliably communicate with the spacecraft. Communications experience with our engineering model was helpful (and essential) but was always much better than we have ever had on-orbit.

6.1.2 Characterization Phase

Our early characterization phase was short (only a few days) and we jumped right into our early mission phase. We did this because we wanted to get at least some mission data down in case we had an early mission loss of the spacecraft. This seemed to impact our spacecraft characterization and a number of spacecraft characterization issues continued for months into the mission: spacecraft attitude, clock correlation and drift, power, etc. Each of these issues (and others) interconnected themselves with our primary mission.

6.1.3 Mission Phase

By late August we were into our nominal mission phase, and were maximizing the data thru our limited communications path. A second ground station was brought on line in November to increase the amount of downloaded data we could collect. In addition, the mission was progressively made more and more automatic and by December we were supporting sometimes as many as 18 contacts a day, without manual intervention. This automation was possible due to reliable communications and spacecraft power, followed by a modular command and control process.

6.1.4 End-of-Mission, End-of-Life Phase

With the loss of our two battery packs, in late January our spacecraft entered its end of life phase. Our tasking dopped off substantially but we continued to collect data for special purposes (2 world wide 10Hz surveys, a southern US lightning signal propagation study with Stanford, a join UCLA, HIPAS, QuakeSat atmospheric experiment, a few 10 HZ earthquake collections, and a small number of California collections related to an earthquake predict from UCLA.) On May 18, 2003 QuakeSat entered it yearly eclipse period, for the next 2 months, QuakeSat will reboot each rev while passing over the south pole. We are looking at options to do occasional tasking during this period as well, but will likely be limited to one or two collections a week. Should QuakeSat survive through the eclipse period additional collections will be made then. As of this writing QuakeSat had

most recently collected data over Iran on May 29 0250z. giving a mission life, to date, of 331 days just under 11 months.

6.2 Mission Reliability

Reliable communications, fairly stable spacecraft clock, and a mean time between reboots of over 2 weeks allowed QuakeSat to support a limited but operational-like mission. Over 3000 collections were tasked and/or collected during the 11 month period of operations, to date.

Reliable communications was fundamental to this. Extra transmitter power, as compared to other small spacecraft on our launch vehicle, provided sufficient link margin for most spacecraft contacts. We believe that the biggest factor negatively impacting communications was QuakeSat's antenna orientation to the ground station antenna. This sometimes resulted in poor communications, but in general communications were consistently pretty good from day to day.

The Fairbanks ground station, while having a higher gain antenna than the one at Stanford, suffers from a dead zone (a poor communications region) to the NE and N at elevations below 20 degrees. This may be due to aurora noise as this is generally in the direction of the magnetic pole.

6.3 Recovery

We used orthogonal (independent) mechanisms to recover failed systems and components, such as the following:

- Proactive reboots at the Fairbanks ground station to cure Heisenbugs (bugs that are difficult to replicate).
- Manual reboots were occasionally needed to recover networking hardware.
- Reactive watchdog timer on the satellite reboots the on-board CPU if it fails to detect the heartbeat for 90 seconds.
- DTMF decoder on the satellite allows a radio signal to force a hard reset and reboot.

Note that this is related to the recursive restartability (RR) concepts of Jamie Cutler and his colleagues (http://crash.stanford.edu/).

6.4 Mission Planning

Our mission planning was a simplified version of a larger earth resources satellite system (like LandSat or Ikonos): a modular approach with increasing levels of detail as the planning process progresses.

Each planning cycle followed the same basic flow:

1) Target areas of interest were defined.
2) Satellite accesses to the target were identified.

3) A subset of accesses was chosen for collection (based on current interest/importance and on availability of system resources like communications throughput, on-board memory, power, etc.).

4) Each chosen access was assigned a high-level collection description which would later map to individual commands and command execution times in command generation.

This breakout allowed mission planning to be independent from collection commanding. Numerous commanding changes could be handled throughout the mission as new command scripts or functional changes to the collections were made. Mission planning was also independent from the exact definition of new or changed target areas. At least three full redefinitions of the target set were handled and numerous new targets added or evaluated during the same time period. The mission planning system was simple but flexible, and was able to evolve to support changes required because of changing collection priorities, changing collection interests, changing flight software, loss of hardware, or changed concept of operations.

6.5 Time

This was one area that was not fully developed or tested before launch. We had a basic high-level concept, but some of the details such as clock stability, variable round-trip delay times (over the internet), reboot frequency, etc were not well known prior to launch and so actual flight data was the first time we tried to model the issue.

6.5.1 Time Reference

Telemetry and payload data needs a stable time reference, and we used an internal "jiffy" clock in the CPU. This clock had to be synchronized with ground-based UTC time. The internal clock also had a drift rate with respect to UTC time, so this had to be tracked. Since all the magnetometer collections were time tagged to execute at prescribed times in the command scripts, this clock correlation was critical. Since our primary mission is of an earth resources type, our magnetometer collection times are defined by when we are over an area of interest.

If there had been no ITAR restrictions on the export of a space-qualified GPS subsystem, we would have included GPS for both orbit position and for precise timing, a feature that would make flight operations much easier. As it was, we used NORAD tracking and a commercial propagation and display program (STK) to plan and analyze the data collections.

6.5.2 Time Estimation

Time and drift rate estimation were done by averaging the transmit and receive times for read-clock commands. A "read clock" command is sent to the satellite, and the satellite's response is time-tagged on the ground. This data was collected and plotted to determine a good time correlation (jiffy clock to UTC time) and drift rate of the jiffy clock. The

correlation value (called the "clock key") was loaded on board and was included in the file header of all magnetometer file downloads. This made every magnetometer data file self-contained with respect to time, i.e. the UTC time of the collection was calculable (assuming the clock key was correct and a zero drift rate of the on-board jiffy clock) without knowing the reboot history of the spacecraft. However, to better account for clock drift and correlation time errors, we also tracked our improving estimates, and our data was time corrected accordingly.

As the mission progressed and better understanding and handling of our on-board clock drift was possible, longer planning periods were supported, further reducing the support required for each individual collection.

6.6 Satellite Commands

6.6.1 Satellite Commands Require Care

Care must be taken when commanding the spacecraft. For example, be very careful when sending low level commands or commands that do not have any validation. The difference between "rm /satellite/data/*" and "rm /satellite/*" is not much typing, but could end a mission very quickly. Typos must be protected against, even if only through a quick pre-pass review or a second set of eyes reviewing the command load.

6.6.2 Short Scripts

A number of short scripts were used to support routine and recurring activities (e.g. turning the beacon on and off during ground contacts, getting the file lengths and check sum for file download and verification, etc). These scripts were zero time-tagged, i.e. marked for immediate execution.

6.6.3 Modular Command Hierarchy

A simple modular command hierarchy was used:

> *Mode: (1HX, i.e. Special 10 Hz mode)*
> *Start and End Events: (1HXStart, 1HXEnd, i.e. group of commands required to start or end a collection or other activity requiring commanding)*
> *Command or script calls: (Example: set digital 1 high, or spawn /scripts/qbeaconctl.sh k)*

This hierarchy allowed mission tasking to be separate from the generation of command loads which in turn was separate from the definition of the events and supporting command scripts. The handling of variables, such as start and stop times and filenames, was provided to allow reproducible collections but with different times and names.

6.6.4 Command De-Confliction

Most command de-confliction was done manually by spacing collections at least 10 seconds apart and not allowing overlaps. However, as we pushed the system we ran into this problem more often and with more time and money we would have incorporated this as well.

6.6.5 Command Upload

The changing concept of operations required late changes to the command and scripting software. In particular, the size of command uploads grew an order of magnitude.

6.7 Housekeeping Data

The basic concept for collection of housekeeping data changed throughout the mission. By mid-mission we could choose specific telemetry channels and the sample rate. This reduced the amount of downlink needed to maintain general health and status of the spacecraft. Sampling every 30 seconds was fine for all internal temperatures. Battery voltages and current were fine at 30 seconds for general health trending, but one sample per second (or better) was needed for good power trending, battery usage, and life cycling (we did not acquire very much of this data).

6.8 Pass Operations

Each pass should be planned with a prioritized list of pass objectives: must do, should do, and would be nice to do. Contact time, especially early in the mission, is limited and very valuable. This list will allow you to do as much as you can, without trying to do too much.

Pre-launch, make a plan for the first several contacts, but realize that this plan will change after launch. Your plan needs to be flexible to changes.

7 Data Management and Analysis

7.1 Data Formats

Dependency on a 3rd-party proprietary format for data locks you in and can seriously impair flexibility. There are two more neutral, non-proprietary formats: binary and text.

Binary format is not portable since the spacecraft CPU's binary format is often different than the binary format of the ground station processor. In this case, endian swapping or zero padding can be required to properly handle raw spacecraft files. This introduces risk of data corruption, and must be designed and tested to operate flawlessly.

Text files are typically double the size of their binary equivalents, but are much more convenient. Almost all commercial software packages can somehow read in a text file. Further, text files can be directly inspected without needing a special browser, converter, or uncompression routine. Using a text format places data in the realm of many tools such as Perl, Unix tools (e.g. scripts or grep/sed/awk), and Internet browsers. Simple copy and paste can be used to move data between applications.

7.2 Data Management

Design of the ground processing operation is primarily a data management problem. There are many types of data: mission planning data used to build command loads, payload and telemetry data brought down from the spacecraft, various meta-data, and data from other sources. Flow of these data implies a data organization structure for a particular spacecraft

mission. The data management design must account for initial operations performed on the data (such as formatting or database storage), plus any analysis and reporting carried out later. In the satellite industry, these requirements were often satisfied via custom software and analysis. Now, much better commercial tools have become available for managing and analyzing data. In summary, the resting places of the data, their structure, and the operations to be carried out on them should be determined early.

8 Project Management

8.1 Organization and Team Structure

8.1.1 Collaboration

One of the best ways to capture the best technologies and spread the workload is to form collaborations with other groups. These collaborations can involve universities, corporations, vendors (willing to donate equipment), and cadres of retired folks who want to share their knowledge with new students. QuakeSat was produced as a collaborative effort among a company, universities, students, contractors, consultants, and vendors.

8.1.2 Job Assignments and Agreements

Assignments within the core team of satellite builders must be stated very clearly, and commitments made that the work will be completed, even if it takes longer than anticipated (and it will take longer).

It is advisable that the project plan or memorandum of understanding (MOU) be written and signed by all parties.

Team leaders and subsystem leaders should be selected early. These are the people required to communicate among the other teams and the overall leader to get problems identified, solutions made, and schedules met.

The team should take a hard look at the skill mix required to accomplish the satellite project. If there is a lack of critical skills, attempts should be made to get additional help in the form of "mentors" or consultants that could guide some activities, even if students or inexperienced team members do the actual work.

8.1.3 Leadership Roles

An organization with two leaders, a Mission Operations Director and a Test Director, has seen much success in integrating larger spacecraft. The arrangement becomes particularly important in the final weeks leading up to shipment of the spacecraft. In smaller teams, there is a danger of oversights if the roles are not clearly assigned. The Mission Operations Director is focused on the logistical and mission planning issues that affect operations. The Test Director is focused on the technical goals of the spacecraft's mission and shipping the best spacecraft possible. Resolution of these two poles of concern is a management decision, to be made out of consideration of the many conflicting forces. It is best if these management tradeoffs are reached out of a pool of two focused concerns.

8.1.4 Meeting and Work Places

It is important early to decide where to meet and work. This is not trivial, since meeting daily or several times per week may be the only way to keep things moving. (Meeting once per week is NOT enough). Email is another way to pass information quickly, and the team should have a "team" email master list so all people can stay informed.

The work area should be large enough, and should have the necessary benches and tools to allow work to proceed efficiently.

8.2 Schedule and Budget

8.2.1 Long Lead-Time Items

Use of complex or expensive test equipment (thermal-vac chambers, vibration facilities, RF anechoic chambers) should be lined up in the early stage of the project. Industries and some universities have these facilities and they may be accessed, but on the owner's terms and on their schedules. The project schedule must be flexible to accommodate these constraints.

8.2.2 Resource Constraints and Contingency Planning

With the typical time, resource and budget constraints on a small-scale project like this, it will be impossible to do everything you would like to do. It will be impossible to remove all risk, and in fact will still be a high risk venture.

A clear focus on the primary objective and the necessary requirements/capabilities to achieve that objective are very important.

Flexibility will be key in many areas: flexibility in organization, in scheduling, in people, in the basic design. Reserves in schedule, budget, and temperament will be required and will be used.

The ability to keep the team and individuals focused and on track, but also flexible to changes and refocusing on new priorities will be required.

The most successful projects are the ones that did the best at optimizing the above mentioned items. Sometimes failure in just one area can lead to complete failure of the project or sometimes the failure will be completely out of your control. However in all cases the measure by which you and your team met the above mentioned items will define the highest possible success of the mission.

The schedule of a project is a direct function of the availability of time of its team members, and the availability of the parts and testing facilities. An inordinate amount of time can be wasted in waiting for parts or facilities. Good contingency planning can greatly help to keep the project on schedule. Also, there is an intangible quality that the team requires—the persistence of the team, and its ability to not take "No" for an answer. Obstacles are meant to be overcome, and creative solutions may be required to find the answers and to keep on schedule.

8.2.3 Launch-Related Costs

Since launch costs may be a large item in the budget, the options should be identified and cost estimates made early. The launch providers also need a minimum of a year or more to integrate your mission into the launch manifest.

For a foreign launch, there are several "hidden costs":

- International Traffic in Arms Regulations (ITAR). Since most satellites are considered "munitions" (as of this date-2004), the team must fill out special forms, and have a Technology Assistance Agreement (TAA) before technical discussions can commence with foreign launch providers. A security plan must also be generated to describe how the technology in the satellite will not be "transferred" to the foreign government. Examples exist on how these documents should be generated. This process should be started at the very start of the satellite project since it may take a year or more to complete the review and approval process at the US State Dept.
- Another hidden cost is the use of an international freight forwarder who is familiar with the export laws and can help ship the satellite and ground support equipment to the foreign launch site.
- After the launch, the GSE must also be shipped back to the US to satisfy the ITAR requirements.
- If a P-POD is used for a CubeSat, the P-POD is considered "returned" when it achieves orbit.
- In the case of a Russian launch, there is an import duty that must be paid to the Russian government. This duty should be declared on the value of the parts in the satellite (for student-built satellites), not on the "market value".

QuakeSat mounted on the flight booster adapter fixture

There is the cost of shipping the satellite AND the ground support equipment to the launch center and shipping the GSE back.

There are travel (including per diem) costs for any of the team that follows the satellite to the launch site.

Communication costs can be significant for some launch areas (e.g. Russia), so use the Internet when available, and be careful of excessive phone charges.

If an American launch provider is used (Space-X, Orbital, LM, Boeing, etc.), there may be unexpected "range utilization" charges for the range telemetry and range safety costs.

If the Shuttle is used, there is a huge cost for all the documentation required for manned flight safety concerns.

8.2.4 Russian Launch Schedule

Russian launches will happen on schedule (at least in their "commercial" endeavors such as EUROKOT). Don't be late, or they will ask you to give them a mass model to launch, and

charge you the same price to launch it while you get back in line for the next launch--- and the new launch will then require additional funding.

9 Summary

We have found that successful small (nanosat) satellites can be designed built and flown for under a few million US dollars, quite a bit less if engineering labor costs can be reduced or eliminated, ie using students in a class/group project. We hope that this information and some of our lessons learned can be helpful to others pursuing similar projects