Modeling and Simulation Tools for Rapid Space System Analysis and Design: FalconSat-2 Applications

CIC Stuart Stanton
LtCol Jerry Sellers
USAF Academy Small Satellite Research Center
USAF Academy, CO 80841
(719) 333-3315
C1CStuart.Stanton@usafa.af.mil
Jerry.Sellers@usafa.af.mil

Abstract—The FalconSat Program at the United States Air Force Academy is a small satellite organization in which participating undergraduate cadets design, build, test, and operate small satellites to carry Air Force or DoD scientific payloads. The team is currently designing FalconSat-2 to conduct measurements of plasma depletions in the ionosphere. Necessary to the design process is the development of requirements and behavioral models to analyze and simulate basic operational scenarios and their effects on the major satellite subsystems. This modeling can be used to preview the impact of potential design changes, reduce development time (especially during hardware and software integration and testing), and aid in training by providing operators with realistic simulations prior to launch. Modeling and simulation for the FalconSat Program uses commercial off-the-shelf software tools such as Excel and Matlab/Simulink environments. The paper describes some basic background on the program, followed by detailed discussion of models being developed and their effect on the evolution of the design. The current status of the design is presented as part of the conclusions.

TABLE OF CONTENTS
1. INTRODUCTION
2. MODELING DISCUSSION
3. CONCLUSION

1. INTRODUCTION

The capstone of the United States Air Force Academy Astronautics curriculum is the FalconSat Program. One goal of the program, housed within the Academy’s Small Satellite Research Center, is to give undergraduate cadets the unique opportunity to “learn space by doing space.” The program facilitates cadet development of small satellite mission design through instructor guidance and mentorship. It allows cadets to gain real-world experience with satellite design, assembly, integration, testing, and operations within the context of a two-semester engineering course sequence. A second goal of the program is to provide a useful nanosatellite platform for Air Force and Department of Defense space experiments. Through FalconSat participation, cadets receive the hands-on opportunity to apply the tools developed in a classroom to a real program, ideally preparing them for the situations they may encounter as officers and as engineers after they graduate.

The current project, FalconSat-2, is the third satellite to be developed within the Academy’s program. In October of 1997, FalconGold was placed into orbit by a Delta launch vehicle to determine whether GPS signals would be valid above the GPS constellation. FalconGold relayed GPS data for 15 days prior to battery depletion. Successful operations and data recovery from FalconGold concluded that GPS signals could be used for orbit determination, even beyond the altitude of the GPS constellation. [1]

FalconSat-1 was launched on January 14, 2000, and represents the Academy’s first “free flyer” satellite. FalconSat-1 flew the DoD-supported Charging Hazards and Wake Studies—Long Duration (CHAWS-LD) experiment, designed to measure electric potential created by a spacecraft’s wake to examine how charging varies throughout an orbit. The CHAWS sensor is designed to assess the hazards for spacecraft operations in the wakes of larger bodies. Unfortunately, a power system problem became apparent soon after deployment. Despite repeated attempts to recover the spacecraft by the cadet/faculty operations team, the mission was declared a loss after only 1 month.

Although it is considered a technical failure, FalconSat-1 represents an academic success for the program as cadets participated from “cradle to grave” in a real-world mission with an all too real-world outcome. Cadets designed and built its payload and subsystems, and they were integral in the mission operations from devising operations plans to participating in the launch campaign to launching the passes of a satellite not operating under nominal conditions. Cadets involved with trouble-shooting the anomalies soon after deployment certainly gained deep insight into system functions and operations.

1 U.S. Government work not protected by U.S. copyright.
Being launch schedule driven, FalconSat-1 proceeded directly to a protoflight model without first building engineering models or development tools. While this “faster, better, cheaper” approach can be effective, it has proven lacking within an academic environment, especially as the program turned to the next mission—FalconSat-2. The lessons learned from FalconSat-1 have motivated significant structural change to the program, with the intention of building a program first and a satellite second. Thus, the new approach has been to focus on building up infrastructure, including design and development tools that can serve as a firm foundation to allow the design to evolve steadily over the course of several missions. The FalconSat-2 design effort is aimed at developing a flexible platform that can be readily adapted and enhanced to meet future payload requirements and secondary launch opportunities.

To meet these new, long-term program goals, it has been crucial to establish reasonably flexible hardware and software models to:

- Facilitate system design trade-offs
- Document and communicate the design configuration to the team
- Support subsystem and software developmental testing
- Support operator training
- Troubleshoot problems encountered during operations
- Serve as a basis for future mission planning and design

Models should include both software to simulate the characteristics of satellite systems and non-flight hardware from which to practice system integration and experiment with different configurations. Ideally, software and hardware should work hand-in-hand to develop understanding of a satellite system by system so that we may reduce the possibility of satellite failure after launch. More importantly, this will keep the FalconSat Program a valuable educational tool for cadets. The following discussion summarizes efforts in the realm of software modeling, how it applies to FalconSat-2, and how it is and will be used with hardware in-loop simulation.

2. Modeling Discussion

Software modeling can be divided into two different categories. A requirements model describes basic, static relationships between subsystems and provides feedback on overall system performance (mass, power, communication link, cost, schedule, etc.). A behavioral model, on the other hand, describes dynamic interactions between subsystems in a simulated operational environment, providing specific feedback on the operational performance of the design with respect to orbital mechanics, sunlight/eclipse cycles, payload duty cycles, etc. [2] Although both kinds of models are significantly different, they can be used together to turn system requirements into spacecraft design (see Figure 1).

The importance of both of these types of models in the development of satellites is discussed below. First, however, it is necessary to establish briefly the expectations of any good modeling tool. Any modeling tool should be representative of the actual system, available early in the design process, and responsive, providing answers to design questions on a timely basis.

In general, it is obvious to expect that any model should be representative of an actual system. While this assumption may seem intuitive, the implementation of such an expectation may be difficult, especially when that system’s design is continually evolving. Consequently, inherent in a good model is a flexible interface: the model must evolve as the system evolves. For modeling to facilitate satellite engineering and design, it is imperative that it be sufficiently flexible. This flexibility motivates the separation between requirements and behavioral modeling, as somewhere in satellite design, a significant shift occurs. As system specifications are eventually determined, design moves to determining how those specifications will act. Therefore, for modeling to be useful and flexible, different models must be used to first determine specifications and then apply them appropriately. It is important to note, however, that the integration of requirements and behavioral models is often nontrivial because they typically differ in their software platforms, data types, and other interface parameters.

![Figure 1 - Relationship between requirements and behavior models within the context of spacecraft system engineering](image-url)
Also, for a model to be useful, it must be available early in the design process. If the purpose is to use models to improve the design process, it is only logical that they be developed simultaneously to, or preferably before, actual system development. This is in contrast to a purely operational model that might be developed only after a system's design is complete.

Finally, modeling must be responsive. Specifically, with regard to satellite design, we refer to models of an entire system, as opposed to more simple subsystem models. The requirements and behavioral modeling for FalconSat-2 has the purpose of aiding in the determination of how various subsystems, ordinarily designed and developed separately, will interact together. It is not uncommon to see a training for cadets. This is what makes model-driven design appealing.

In any satellite development process, these relationships must eventually be realized. However the traditional approach of independent subsystem design has made this difficult, and any communication deficiency anywhere throughout the process will invariably slow down the design process. But if models are used to support a more concurrent engineering philosophy, then satellite design becomes more efficient. Ultimately, this is what motivates the use of modeling for FalconSat, primarily due to the unavoidable time constraints stemming from the requirements of undergraduate education and military training for cadets. This is what makes model-driven design more attractive than the traditional document driven approach: a responsive model that gives answers in days, for example, instead of in weeks or months, is far more appealing.

Rapid system development is an obvious need for commercial satellites due to the natural time constraints imposed there. But the benefits of a quicker process can also be realized within an academic environment, especially at the USAF Academy. As the design team consists of cadets who must also pursue the breadth required for an undergraduate degree and receive the training required for military officers, time is a premium. On average, students can only devote less than 10 hours/week to the project.

Of course, there are other ways to speed up the design process without resorting to extensive modeling and simulation. We can appropriately compare ourselves to the SNAP program at the University of Surrey, as FalconSat-2 will use several components developed for SNAP. Surrey Satellite Technology Limited (SSTL) developed this nanosatellite platform (6.5 kg in mass) in an extremely short amount of time (only 9 months from design to launch). Launched in June 2000, this capable nanosatellite mission has proven to be a great success. [3] However, while significant system testing was conducted on the ground, the team made little use of extensive simulations. This was only possible because the team consisted of seasoned engineers and operators who could build upon dozens of orbit years of operational experience.

In contrast, student satellite programs, such as the one at the USAF Academy, begin with virtually no previous experience. To make matters worse, during a multi-year development, there will be a nearly 100% turnover of students participating in the project. Thus, the program has turned to model-driven system design to replace the older document or requirement centered system design. David B. Smith at JPL describes this sort of engineering: “The model environment eliminates over specification, establishes real concurrent communication, and links early prototyping to actual testing of the flight hardware. Continuous verification and validation of the design is now possible thorough this approach and reduces the Systems Integration test time, at the end, by a factor of two.” [2]

The concept of model-driven design, although still quite new, has been the subject of much attention recently, especially at the Jet Propulsion Laboratory. Our efforts at the USAFA Small Satellite Research Center have primarily been based on their research into reinventing the design process, progressing to concurrent engineering based on design models. JPL’s Team X concept, motivated by NASA’s direction to control and reduce operations costs, has resulted in a more efficient design process, asserting that “the best way to reduce operations costs is to consider operational issues early in the design process, of equal rank with spacecraft and trajectory design issues.” [4] This has motivated the development of a FalconSat requirements model that is effectively a scaled-down version of the tools employed by Team X.

Requirements Modeling

The first order analysis of establishing requirements and specifications for a satellite follows the basic evolutionary steps shown in Figure 2. A comprehensive requirements model combines all these steps into a single, integrated software tool.

The primary purpose of a requirements model is to convert requirements into specifications. It first provides the opportunity to identify requirements, on the mission, system, and subsystem levels. It offers the ability to track requirements and how they flow into subsystem specifications, allowing for requirements and their changes to be reflected in any first order analysis. More importantly, though, a requirements model gives a designer the opportunity to analyze trades between subsystem specifications, as the relationships between subsystems established in the model allow for one to see the effects of one specification or requirement of one subsystem on the
specifications or requirements of another subsystem. Consequently, this sort of modeling allows system designers to determine basic system performance budgets like power, data, and communications, and it allows them to develop an intuitive understanding of the basic trade-offs between each for a given system.

For Falconsat-2, requirements modeling and analysis has been conducted using an Excel spreadsheet that establishes subsystem relationships by applying the first order tools described in *Space Mission Analysis & Design* (SMAD). [5] We began by modifying a generic design analysis tool developed by Dr. David Cloud to complement the SMAD reference book. [6] Figure 3 shows the basic interface provided to a user to begin this first order analysis.

From this interface, users can begin to enter basic design parameters including known or fixed subsystem specifications, orbit or other constraints, and payload support requirements. From there, performance budgets, and associated margins, can be quickly determined to identify potential problem areas in the design before the consequences of those problems become significant. By providing instantaneous answers to several immediate design questions with regard to subsystem interface, the requirements model has reduced a process that could take months down to several weeks.

Electrical Power Subsystem analysis provides a good example of the how the requirements model is being used. Preliminary estimates of subsystem power requirements are first input into the *Sizing Summary* worksheet shown in Figure 4. These estimates are then combined into the required spacecraft power on the *Solar Array Sizing* worksheet shown in Figure 5. This worksheet also includes basic assumptions about array design including lifetime, efficiency and size constraints. Shortcomings of the preliminary array design can then be identified back on the *Sizing Summary* worksheet and the engineering team can then begin to iterate the current versus preliminary estimates until the solar array design achieves positive margin.

The requirements model used for Falconsat-2 has proved to be an invaluable tool for the design team. With a fairly user-friendly interface, areas of concern immediately present themselves to the designer. For example, early in Falconsat-2 development, problems with the downlink and payload data rate were noticed, allowing for quick resolution with limited consequences to other subsystems because of this early detection (see above).

However, as one may see in Figure 3 above, other input options exist to achieve similar results: the values calculated here go into a System Sizing Summary that allows the designer to coordinate sizing requirements of all major spacecraft subsystems concurrently.
FalconSAT-II Design Sheet Navigator

**Orbit Analysis**
- Orbit dynamics
- Mission geometry
- Orbit maneuvers and maintenance
- Delta-V & geometry budgets

**MESA Payload Analysis**
- Launch Vehicle Information
- Launch Vehicle Reqs

**Spacecraft Subsystems**
- Power
  - Solar array analysis
  - Secondary battery analysis
- Communications
  - Uplink
  - Downlink
- OBC
  - Specifications

**MUSA**
- ADCS
- Torques
- Strings
- Alt - Torques
- Alt - Strings

**System Sizing Summary**
- Sizing Summary

---

**Fundamental Budgets**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>11.31 kg</td>
</tr>
<tr>
<td>Peak Power Budget</td>
<td>11.53 W</td>
</tr>
<tr>
<td>Ave Power Budget</td>
<td>6.49 W</td>
</tr>
<tr>
<td>Uplink Margin</td>
<td>34.21 dB</td>
</tr>
<tr>
<td>Downlink Margin</td>
<td>-2.51 dB</td>
</tr>
</tbody>
</table>

**Design Parameters**
- Mass margin percentage: 10.0%
- Power margin percentage: 10.0%

**Preliminary Estimates**
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Mass (kg)</th>
<th>Peak Power (W)</th>
<th>Ave Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload</td>
<td>0.8</td>
<td>0.06</td>
<td>0.03</td>
</tr>
<tr>
<td>OBC + SIM</td>
<td>8.7</td>
<td>1.0</td>
<td>0.10</td>
</tr>
<tr>
<td>Power</td>
<td>2.5</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Comm Tx</td>
<td>0.6</td>
<td>3.3</td>
<td>0.7</td>
</tr>
<tr>
<td>Comm Rx</td>
<td>0.55</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Thermal</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>ADCS</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Structure</td>
<td>3.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Margin</td>
<td>0.9</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td>Totals</td>
<td>10.4</td>
<td>11.3</td>
<td>6.5</td>
</tr>
</tbody>
</table>

**Current Estimates**
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Mass (kg)</th>
<th>Peak Power (W)</th>
<th>Ave Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload</td>
<td>0.8</td>
<td>0.00</td>
<td>0.04</td>
</tr>
<tr>
<td>OBC + SIM</td>
<td>9.5</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>OBC</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Power</td>
<td>1.3</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Comm Tx</td>
<td>0.9</td>
<td>3.3</td>
<td>0.7</td>
</tr>
<tr>
<td>Comm Rx</td>
<td>0.33</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Thermal</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>ADCS</td>
<td>0.6</td>
<td>4.2</td>
<td>2.5</td>
</tr>
<tr>
<td>Structure</td>
<td>4.9</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Margin</td>
<td>1.0</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td>Totals</td>
<td>11.3</td>
<td>11.3</td>
<td>6.5</td>
</tr>
</tbody>
</table>

---

Figure 3 - User interface for the FalconSat-2 Requirements Model

Figure 4 - Sizing summary for the FalconSat-2 Requirements Model

7-3483
The requirements model is useful in other ways, as well. Most importantly, the requirements model is the medium through which engineers and managers can communicate. It serves as the focal point for various subsystem managers to input their requirements and specifications. Through this model, then, design members can maintain an up-to-date understanding of system development by simply accessing the portions of the model where managers or specialists record their results. But it is the next step that the requirements model takes that makes it so attractive. Any documentation method can be used to update requirements and specifications, but a requirements model takes these updates and determines how these updates effect other subsystems. This is crucial, as a subsystem designer does not have to seek out information about other subsystems because what statistics he needs are automatically updated for him. If used properly, a requirements model avoids the difficulties that arise with ordinary communication methods.

This implies, then, that the requirements model can replace ordinary specification documentation. Preferably, the sort of model will serve as the primary documentation source for program design information. Ideally, then any sort of program update, whether a briefing or a report, can consist primarily of material copied identically from the requirements model. The appeal to this concept is that it avoids the tendency in some project development toward “PowerPoint Engineering,” where information is only documented when it is to be presented in a briefing.

Finally, the requirements model can serve as a first-step tool in “what if” analysis. This offers the interface to look at the effects that result from specification or requirement changes or the introduction of different technology options. This sort of analysis will become important in behavioral modeling as well, but at this level, one may assess quite rapidly the potential consequences, good or bad, of minor to major design changes. As the requirements model establishes and manages mass, power, data, link, and other fundamental performance budgets, one will be able to see how these are affected by choosing different design options.

Requirements modeling can be useful through several iterations, as subsystem requirements or specifications are modified until it becomes apparent that overall program objectives will be met from all aspects of the mission design. The primary limitation of an Excel-based requirements model such as this is the static nature of subsystem interaction, especially with the dynamic space environment. Thus, after the basic spacecraft configuration has converged with the aid of the requirements model, it is necessary to apply these specifications to behavioral modeling to understand these more subtle, second-order effects.

Figure 5 – Solar array sizing interface for the FalconSat-2 Requirements Model
Behavioral Modeling

The behavior model helps to understand how specific subsystem configurations or payloads act, over time, in the actual mission environment. Behavioral modeling offers the possibility of simulating an entire spacecraft mission—to include nominal to worst-case scenarios—prior to spacecraft launch. Several advantages are derived from this. Specifically, satellite operators have a training tool by which they may see most, if not all, possible mission scenarios before having to react to them in real time. More importantly, however, and the primary purpose for our behavioral modeling, is to speed up the development process.

But the FalconSat process is not only shortened but also improved by implementing models that simulate the behavior of integrated satellite subsystems. The assembly, integration, and testing for FalconSat-2 is crucial for a successful mission. This is an intuitive point, but suspected subsystem failures in FalconSat-1 motivate this emphasis. By integrating simulation to the FalconSat testbed, the model results can be compared to flight system results to validate each of the hardware components.

Finally, the behavioral modeling for FalconSat provides a workable user interface appropriate to offer simulations for satellite operator training. Specifically, systems may be altered or hampered within the modeling environment to offer many operational scenarios beyond the nominal. Examples include failing subsystems or inaccurate telemetry.

We have placed significant effort into developing behavioral models of spacecraft subsystems in order to apply the advantages of model-based design to FalconSat-2. Dynamic models have been developed in Matlab to serve this need. Simulink, a graphical modeling environment in Matlab, has provided an appropriate interface for time-based simulation of the FalconSat-2 system, deliberately organized by subsystem. The figure below shows the model.

The behavioral model in Figure 6 is founded primarily upon two subsystems, Power and Attitude Determination and Control (ADCS), and the orbital environment they will face. These are the subsystems that have perhaps the most potential influence on satellite operations, and consequently, the modeling efforts for them have been the most extensive. For example, all active subsystems require interaction with the power system to be used. But the satellite’s ability to provide power will be primarily dependent upon orbital conditions (satellite location in orbit relative to the sun) and attitude conditions. These primary relationships seen over a scenario time are vital to determining the ability of the satellite to perform its mission. After this, relationships between these two subsystems, the satellite environment, and other satellite considerations like communications and thermal activity can be modeled.

![Figure 6 - FalconSat-2 Behavioral Model in Simulink](image-url)
The significance of this sort of modeling is seen in how we intend to use it in the FalconSat environment. Here, one may see how software modeling can be connected to hardware modeling in simulation. To complete the loop, then, the design of FalconSat-2 has included building a desktop laboratory of prototype hardware for the primary spacecraft subsystems. FAST, the FalconSat Avionics Simulation and Testbed, is our hardware-in-the-loop modeling environment. Using this environment with the behavioral models developed in software may prove to be the most useful aspect of modeling. This may happen in two ways.

First, hardware may be put through scenarios that can be simulated in software to ensure that hardware eventually intended for space flight, is operating as it should. The intention here would be to compare an entire system of hardware to an entire system of software. More exciting, however, would be the ability to test particular subsystem hardware by integrating it with the software dynamic models. This is why behavioral modeling has been in Matlab and why that modeling has identified the contributions of each subsystem specifically. Using the data acquisition capability in the Matlab environment offers the possibility of testing a single piece of hardware against a software simulated environment, including the contributions of an environment and the interactions with other spacecraft systems to understand how an individual subsystem, and eventually the entire system, will react in space. By understanding this, we will be able to recognize areas of concern early and compensate for them before a problem occurs.

To illustrate a real-world capability of behavioral modeling, then, we provide an example of data received from the Power portion of the behavioral model for a given scenario. Specifically, the FalconSat-2 behavioral model is run in a low earth orbit for several orbital periods at nominal power requirements (primary subsystems running in their nominal states for the duration). Below is shown basic telemetry from the power system: solar array power for four solar arrays, the battery voltage, and the battery current.

![Figure 7 - FalconSat-2 Behavioral Model Power Telemetry—Solar Array Power, Battery Voltage, and Battery Current vs. Time](image-url)
These results are verified in two ways. First, because subsystem states are not changed during the simulation, one would expect that the output would be periodic, as it is. However, the more important validation of the results comes from the results of SNAP-1, as the power system for Falconsat-2 is identical to that of SNAP. Specifically, the battery current data, indicating rapid oscillation between positive and negative current through the battery when it is fully charged, matches quite well with SNAP results. These indicated the sharing of the power load between a charged battery and the battery charge regulators that convert solar array power to useable current.

3. CONCLUSION

The FalconSat Program in the USAF Academy's Small Satellite Research Center faces many challenges in its effort to build useful space platforms within the confines of an undergraduate institution. Time, turnover, documentation and educational needs have lead the team to turn to both requirements and behavior models to achieve an effective design in the shortest possible time.

The FalconSat-2 program has been divided into the delivery of four systems between the Fall of 2000 and the Spring of 2002. Discussed earlier, FAST is the hardware testbed for FalconSat-2. FAST was assembled in November 2000, and at the time of this writing was prepared for initial testing. An engineering prototype of FalconSat-2 will be deliverable no later than April 2001 for rigorous environmental testing. A qualification unit and flight model will be developed during the 2001-02 academic year.

Our modeling efforts will thus be the primary resource for rapid FalconSat-2 analysis and design. Currently, the requirements model is approximately 80% completed and is being used on a daily basis to assess the impact of design modifications in solar panel size and efficiency and link budget analysis. The behavior model is approximately 70% complete and initial results with power subsystem modeling have correlated quite well with on-orbit data from a spacecraft with an identical design. A full-up hardware-in-the-loop simulation and testing environment has also been established is currently being used for software development and payload integration and testing.

In the coming months, the team plans to use and develop all of our modeling and simulation tools to facilitate the design, assembly, integration and environmental testing of an engineering model for the FalconSat-II spacecraft. These tools will be absolutely necessary for us to effectively "fly before we build it." [7] We are confident that these models will be useful tools for documentation, analysis, troubleshooting and providing in-depth understanding of the system design for the students to help speed us toward our ultimate goal of the launch of FalconSat-2 as a secondary payload on the Space Shuttle in late 2002.

REFERENCES


Stuart Stanton is a first class cadet at the U.S. Air Force Academy, preparing for graduation and commissioning in May 2001. Majoring in Astronautical Engineering, he is the cadet program manager for the design of FalconSat-2.

Jerry Jon Sellers is an active duty Lieutenant Colonel in the U.S. Air Force. His work experience includes: Guidance & On-board Navigation Officer, Space Shuttle Mission Control Center, NASA, Johnson Space Center, Houston, Texas; Assistant Professor of Astronautics at the USAF Academy; and Chief, Astronautics for the Air Force European Office of Aerospace Research & Development, London, UK. His educational background includes a BS in Human Factors Engineering from the USAF Academy, an MS in Physical Science/Astrodynamics from the University of Houston, Clear Lake, an MS in Aeronautics/Astronautics from Stanford University and a Ph.D. in Satellite Engineering from the University of Surrey, UK. Currently he is the Director of the USAF Academy Small Satellite Research Center in Colorado Springs, Colorado.