SSC23-WII-06

Development and Implementation of Automated Planning in CubeSats

Liam Riley, Vedika Ghildyal, Cameron Bonesteel, Deepak Mishra
University of Georgia, Small Satellite Research Laboratory
220 Cedar St. Room 107 Athens, GA 30602 ; (252) 732-6979
William.Riley@uga.edu

ABSTRACT

The University of Georgia's (UGA) Small Satellite Research Laboratory (SSRL) is developing two CubeSat missions, each with ambitious operational goals that require accounting for a large number of variables. Given the inherent difficulty of fulfilling these requirements without complex onboard autonomous decision-making software, an automated ground-based solution is desirable. To meet this demand, SSRL is implementing an operational pipeline to effectively integrate the newly authored Multi-aspect Automated Satellite Scheduler (MASS) into both missions. MASS is administered and tested by SSRL's Mission Operations and Flight Software teams. MASS implements a complex decisional tree informed by CubeSat operational requirements, and determines operational schedules while accounting for internal and external variables. SSRL's Mission Operations team will analyze and verify output schedules to prevent procedural errors or unforeseen anomalies. SSRL's Flight Software team will utilize ground station software and pipelines to automate conversion of output schedules to satellite-readable formats ready for uplink and automate downlinking telemetry to the MASS program to improve prediction accuracy. This operational pipeline and the MASS program are intended to be mission-agnostic and available to all CubeSat operations.

INTRODUCTION

SSRL's Multi-View Onboard Computational Imager (MOCI) is currently in the later phases of development, making proper functionality of its mission operation systems increasingly crucial. To that end, all opportunities to decrease operator effort to access critical satellite health telemetry must be utilized. The Mission Operations team must be able to retain full focus on the successful retrieval of mission critical data, while also carrying the responsibility of training operators to effectively respond to both nominal day-to-day operations and anomalous scenarios.

Mission Operations is the most critical part of maintaining a satellite over its lifetime. Operations must be meticulously planned during pre-launch phases, and tolerances only become more stringent post-launch, where a strong understanding of spacecraft behavior, ground system functionality, and the mission operations plan must be established in order to achieve a successful mission. The most realistic response to these demanding tasks, when given human error, is an automated solution. In the case of MOCI, and more generally for any CubeSat that does not contain complex autonomous decision-making software but has an ambitious and extensive operational concept, the solution is automated predictive scheduling. To meet this demand, SSRL is implementing an operational pipeline to effectively utilize the newly authored Multi-aspect Automated Satellite Scheduler (MASS). The MOCI implementation of MASS is shown as a block diagram below in Figure 1.

In the following sections, each step of this operational process will be outlined in detail. First, we will discuss the development of MASS as an implementation of complex decisional trees as informed by an operational concept which determines operational schedules while accounting for a variety of internal and external variables such as orbital parameters, power generation, and data handling. Second, we will discuss the integration of MASS, wherein the Flight Software team utilizes ground station software and pipelines to allow for automated conversion of output schedules to satellite-readable formats ready for uplink and downlinking of telemetry to further inform the model. Finally, we will discuss the verification of MASS, wherein the Mission Operations team utilizes an operational pipeline to analyze and verify output, precluding unforeseen anomalies with the use of general operators to provide 24-hour presence despite the scholastic requirements of a college environment, flight directors to provide detailed subsystem advice, and a mission director to handle final schedule approval.

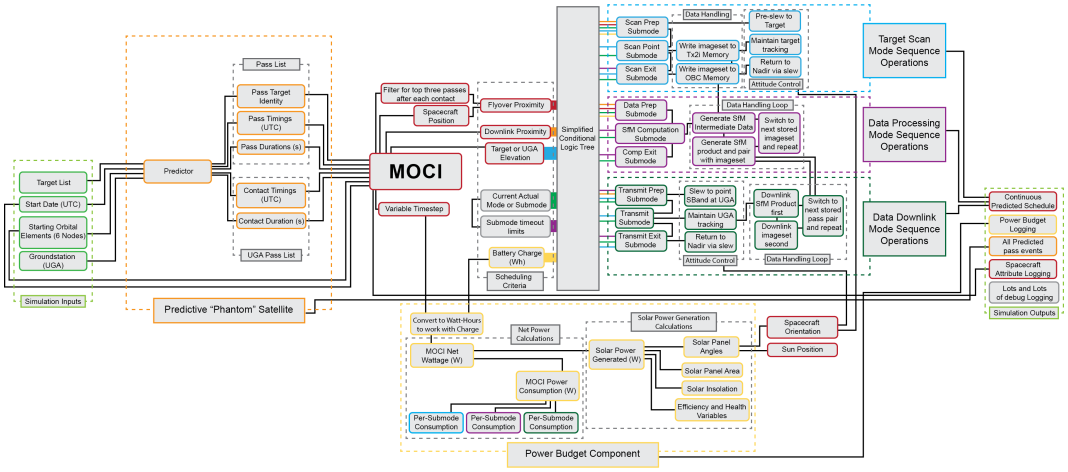

Figure 1: A block diagram displaying a broad overview of each component in MASS's MOCI implementation, as well as the relation between each component

Crucially, this operational pipeline and the MASS program can be configured for all CubeSat operations, allowing for the simple management of complex and ambitious missions by teams with limited resources.

RELATED WORKS

Satellite automation and autonomy has been a long-standing challenge in CubeSat development that has had a variety of solutions over time.[1–8] Automation is different from autonomy in the level of control the program has over itself. Autonomous systems tend to have a higher level of intelligence and therefore less of a need for human intervention than automated systems. As a result, autonomous systems are more difficult to implement on CubeSats due to the computing power often available being relatively low compared to ground-based computers. Thus, an automated approach with ground-assistance seems to be ideal and has had several different approaches over the past couple decades.[1–4,6]

Some of the simpler approaches involve the ground system finding an ideal schedule for the satellite. Grasset-Bourdel et al. proposed a chronological forward search algorithm to accomplish this.[1] Their approach is developed using priority and decision levels that work together to produce a schedule that is not necessarily ideal but rather simply viable for satellite execution.[1] This approach is the closest to what will be proposed in this paper.

Ground observation satellites such as MOCI prove a particularly difficult challenge. Chu et al. discusses the difficulty of scheduling Earth observation satellites due to their need to be over a target to observe it.[2] In response, they propose an onboard scheduling system that searches for the next observation during the current one as a way to maximize target opportunities.[2] All of these systems have something in common; they require extensive sets of parameters for navigational and satellite health data to make decisions in complex decision trees or algorithms.[1–3,6–8] Some systems such as the one proposed by Baek et al. even consider variables outside of satellite navigation and health, such as deadlines and monetary benefit to the end user.[3]

While these ground-based approaches are common and thoroughly documented, autonomous approaches are much fewer and far between. Wörle et al. proposed an onboard near-autonomous approach that is assisted by ground-based systems to provide the satellite with more data for decision-making.[4]

The solution proposed in this paper is an automated approach meant to be a best fit for the SSRL's missions while also being scalable to new missions. We hope that this work along with some of the other's cited here will lead to fully autonomous solutions in the future.

MASS DEVELOPMENT

MASS began as an effort to create a power budget simulation for SSRL's MOCI CubeSat, first using ANSYS's Systems Tool Kit (STK), then moving to FreeFlyer. Soon afterwards, it became clear that FreeFlyer had far more capabilities than simple power budgeting, and could be used to simulate not only all of MOCI's budgets, but also future event timings. With this understanding, MASS began to take shape as a comprehensive predictive scheduler.

Functional Overview

MASS has a large number of different components, overarching functional blocks of code, that work in concert to properly schedule a CubeSat. Mode simulations recursively affect the power, data, and link budgets that in turn inform which mode may begin. The major components that MASS requires to function – pass prediction, budgeting, and mode determination – will be detailed here.

The distinct advantages FreeFlyer has over other mission planning softwares available to SSRL will also be detailed to illustrate the decision for MASS to be written with it. FreeFlyer's ease of automation and operator use has made it invaluable, and allowed for a quick development period that has already allowed for preliminary results and procedure validation of MASS, a showcase of which is also included here.

Pass Prediction

One of the first essential components of MASS is its ability to propagate and predict orbital motion and important events, chiefly ground station passes. This is accomplished through a duplicate "phantom satellite" that begins at the present position of the simulated CubeSat and is then propagated forward over the course of the designated simulation time while the CubeSat proper remains still, time-wise; this is illustrated in Figure 2.

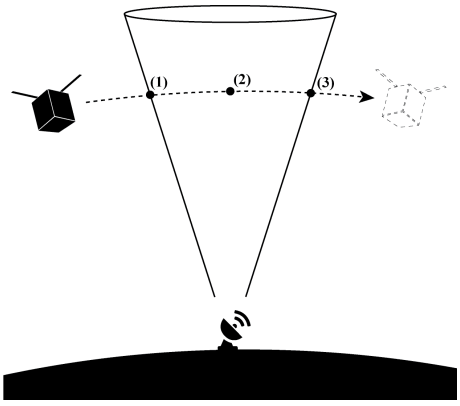

Figure 2: The propagation of a "phantom" satellite that records the timing of (1) entering line of sight, (2) peak elevation, and (3) exiting line of sight

Over this time period variables relating to various schedule-related events are collected, including the precise timings of entry and exit of ground station and target lines of sight (LOS), and dangerous regions such as the South Atlantic Anomaly. MASS can also collect data such as the highest angular elevation the satellite achieves with respect to a target or ground station. MASS can also be configured to perform predictive event monitoring for many other events. Other applications include, but are not limited to, closest passes or loss of signal with other CubeSats, as in a satellite swarm.

MASS propagates a phantom satellite in order to gather a table of all predicted future events which must be referenced in order to properly determine modes. Mode transitions often require forewarning in order to turn on satellite hardware before an event, slew the CubeSat to a predetermined attitude, or preemptively activate safe mode.

Upon completion of the prediction run, the phantom satellite is deleted, and the CubeSat begins to propagate along the path "scouted" by the phantom. This ensuing main simulation is much more extensive and models the components concerned with simulating the time-dependent state of the CubeSat such as each of the budgets. The combination of these budgets and the previously obtained list of predicted events allows MASS to produce predictive schedules, but is more computationally expensive due to the added complexity, and as such the prediction can run up to a hundred times quicker than the main simulation without error.

Power, Data, & Link Budgeting

MASS endeavors to properly simulate the state of a CubeSat's major software and hardware budgets over the course of the simulation. The most important of these states is the power budget, which must be balanced or else risk critical mission failure. In MASS the power budget coincides with the pointing budget, as most CubeSats have fixed solar panels which are insolated based on the attitude of the CubeSat. To this end, MASS first calculates a set of variables described in Table 1 and combines them via equation (1) to produce gross wattage. This equation omits some behaviors of solar panels, such as proper current curves, in favor of computational simplicity but serves nonetheless as an effective boilerplate function for calculating power generation. Upon generating this figure, MASS then subtracts the power draw of the CubeSat hardware as per provided power figures and computes the net gain or loss in watt-hours, which is added or subtracted from the overall state-of-charge.

Table 1: Power Simulation Variables

Variable	Symbol	Description
Mean Power (W/m^2)	G	The average solar insolation at a distance of 1 Astronomical Unit (AU)
Solar Distance (km)	$d_\odot$	The distance of a satellite from the Sun
Panel Area (m^2)	p	Each separate solar panel's square area
Vertex Angle (radians)	θ	The angle between a solar panel's normal vector and the Sun
Solar Degradation (%)	a	The percentage extent of a solar panel's loss of capability
Efficiency (%)	ν	User-set constant value of the solar panels' efficiency
Astronomical Unit (km)	AU	Earth's average distance from the Sun

$$\text{Gross Wattage (W)} = a\nu p|\cos(\theta)|\frac{Gd_\odot}{1AU}, \quad (1)$$

MASS additionally simulates data and link budgets for satellites with uplink and downlink requirements. MASS can be quickly configured to generate large arrays and populate them with arbitrary figures representative of the file size of any given data product on board a CubeSat. By changing the contents of these data arrays, MASS can simulate the reading and writing of data as it would occur during any given data processing mode, such as the gathering and storing of telemetry onboard a CubeSat. This data can also be summated and compared against a maximum hard-drive storage value to avoids overwriting important data products and directs CubeSat behavior accordingly by rescheduling modes. During mission operations, these arrays can be referenced on the ground to gain an understanding of what products remain on the CubeSat. The link budget is similarly informed. When the CubeSat is scheduled to contact a ground station, MASS may simulate the downlink and uplink of data such as telemetry or mission products from the CubeSat to ground control via array manipulation constrained with maximum downlink speeds and write speeds, while accounting for random fluctuations in these speeds.

Mode Determination

MASS is capable of tracking a large number of variables, such as the aforementioned state-of-charge, data capacity, CubeSat attitude, and up/-downlink, alongside the predicted events list. These variables are influenced most directly by modes and sequences of modes, which MASS extensively models. MASS is primarily designed to handle modes in the context of a finite state machine (FSM), which consists of a series of discrete states corresponding to operational modes, with conditional rules governing transitions between states. In MASS this concept is realized as mode "sequences" with a series of submodes. Once a mode is allowed to initiate, an "on-rails" set of submodes are executed in sequence. MASS is by default configured to have a central mode, "Cruise mode", which is stable and power-positive. Mode sequences may only be entered from this central mode and no other mode sequence may be entered during the duration of an ongoing mode sequence, shown in Figure 3.

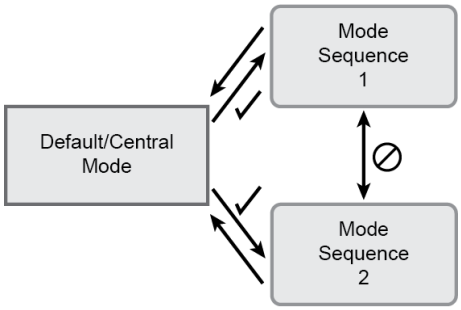

Figure 3: A diagram displaying MASS mode transition rules

Mode sequences can accomplish an arbitrary set and number of actions in MASS, but in general consist of a per-hardware set of power draws that together affect the battery state-of-charge, a data management component for software processes during the mode, and sometimes a pointing and link component to cover attitude maneuvers or ground station contact.

In MASS this FSM is implemented as a –structurally identical– behavior tree, outlined in Figure 4. The model is that of a behavioral tree because MASS is designed to cater to operationally complex CubeSats, and Finite State Machines begin to become convoluted beyond five to six states. MASS takes in a set number of specified variables at each time step of the simulation and compares them to a configurable set of criteria equivalent to the CubeSat's operational rules. MASS then attempts to create as intense a schedule as possible by planning a mission-relevant mode whenever variables exceed safe thresholds, e.g., when state-of-charge is high enough to allow a power-hungry processing mode.

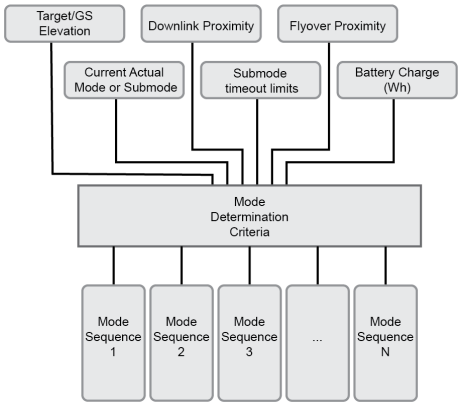

Figure 4: An overview of mode determination in MASS, with a small subset of the actual input variables

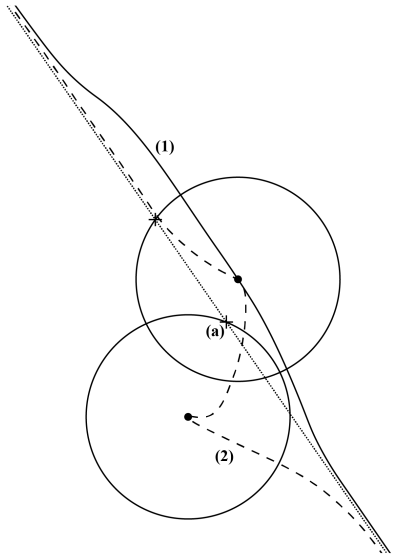

Figure 5: A model of the optical path of the MOCI satellite as it passes two overlapping target zones with (1), and without (2), target priority, with the "hot swap" error point where MASS would otherwise attempt to switch targets noted as (a)

Mission-Specific Adaptability

MASS allows for extreme customization at all levels to augment its general functions. While at base it is a useful tool for CubeSat automation, MASS cannot automatically account for each unique potential operational concept presented. As such, in addition to setting initial variables such as component power draws, solar panel configurations, or mode specifications, more in-depth changes must be made. This has been the case with the two missions MASS has been adapted for at the SSRL. Each of these two missions required special functions in order to properly simulate their unique concepts. This process will likely be necessary for most applicable CubeSat missions, but simple to implement due to MASS's adaptability.

MASS was first applied, and modified, to the Multi-View Onboard Computational Imager (MOCI). A complex target priority system was implemented in the prediction component of MASS to prevent MASS from commanding erroneous "hot swapping" of targets as the simulation detected new target flyovers, as shown in Figure 5. Instead, each period between two contacts with the UGA ground station is portioned out, and the three target flyovers within each period with the longest flyover time are selected, provided they do not overlap. In this manner, the simulated MOCI CubeSat may smoothly focus on pre-slewing to a single target at a time.

MASS was subsequently applied to the Mission for Education and Multimedia Engagement Satellite (MEMESat-1). MEMESat-1 uses an ambitious passive magnetic attitude control (PMAC) system, which orients it via nutation rods along Earth's magnetic field lines. This poses unique challenges for MASS which must adequately simulate MEMESat-1's attitude over time. This was accomplished by creating a database of the magnetic vectors at MEMESat-1's altitude for each and every latitude and longitude and referencing this to point MEMESat-1 along an appropriate attitude. Each of these added functionalities were fairly simple to add to MASS's code due to MASS's use of discrete functional code sections, which reduce the danger of code interaction errors. These demonstrate MASS's applicability to arbitrary CubeSat operational concepts. This is mostly thanks to MASS's parent software, FreeFlyer, which provides a conducive language and environment for the simulation.

FreeFlyer Flight Dynamics Software

The open source FreeFlyer program has been invaluable to creating the MASS software because of a

series of advantages that favor FreeFlyer over a number of similar programs such as MATLAB, SPICE-wrapped Python or Cosmographia. FreeFlyer is an independently verified and validated software used for spacecraft analysis and operations by NASA, NOAA, USAF, NRO, and commercial satellite providers, and is the program of choice due to the following factors:

Ease of Use

A primary benefit of FreeFlyer, and frankly the swaying factor in most situations for university-led CubeSat missions, is the FreeFlyer University program, which allows university students to download FreeFlyer at no cost. This free software is not diluted in any way and contains every feature and capability of the highest paid version used in the aerospace industry. This is obviously highly desirable for university aerospace initiatives which often operate on very small budgets that may not be able to afford mission operation programs at their full pricing.

The FreeFlyer's open-source software runs on a unique scripting language designed to facilitate orbital simulations by default and revolves around preset space environments to minimize creating functions "from scratch". The language focuses on astrodynamics applications and ensures users don't have to wade through non-relevant functions and calls to find what they need. The language structures code in a 'mission sequence' composed of code blocks, providing simple modularity which has facilitated MASS's various functional components. This software provides MASS operators with the freedom to change arbitrary input variables to customize the simulation and edit the output variables for detailed debugging and verification of functionality.

FreeFlyer has many features, but as with any program has certain areas it may not be as suited for as other programs. In these instances another program may need to be linked to MASS in order to properly simulate certain CubeSat processes. FreeFlyer has a robust runtime API allowing it to either call external programs during FreeFlyer mission scripts or be called during the process of another external program. In this manner, FreeFlyer can be effectively integrated into operational satellite ground systems, which is a key goal in the process of implementing the MASS software.

Simulational Fidelity

MASS's success criteria are linked nearly entirely with whether it can accurately match and predict the satellite during its post-launch. When producing schedules with accurate timings, the primary area of introduced error is in orbital position. The CubeSat must be propagated for multiple days and hundreds of orbital periods with less than a kilometer of positional error, which would shift timings for mission-critical modes, potentially leading to failure to accomplish mission objectives. FreeFlyer has a spectrum of integrators that allows for a nuanced decision between computational burden and simulation accuracy. Preferentially, MASS should be run on highly capable hardware, namely modern commercial CPUs or stronger, to reduce simulation run times, as MASS's default integrator is the fourth order Runge-Kutta method, whose accuracy is illustrated in Table 2, and whose computational burden allows for an acceptable run time speed. For further accuracy, FreeFlyer allows MASS to easily link to ephemeride files and programs such as SPICE.

Table 2: FreeFlyer Integration Methods[9]

Method	Accuracy	Speed	Usage
Runge Kutta 8(9)	10^7x	1x	Interplanetary design
Runge Kutta 4(5)	10^6x	2.2x	General orbit propagation
SGP4	10^5x	10x	Short-period stable propagation
Cowell	10^3x	5x	Multiple spacecraft
Bulirsch Stoer	10^2x	4x	Orbital decay analysis
J2Mean	10x	10x	For early mission concepting
Two-Body	1x	20x	For low-accuracy visualization

For CubeSats where positional accuracy is not the primary determinant of schedules, MASS's stringent requirements in this area can be relaxed, and other areas may be prioritized for high fidelity. As aforementioned, MASS can be linked to a wide variety of other programs via its run time API. One of the best uses of this is to link with thermal simulations, given the particular risk of overheating to CubeSats without dedicated radiators. Lastly, MASS can be specifically configured to cater to mission-specific accuracy concerns, as aforementioned in the case of MEMESat's PMAC systems, which could not be easily simulated in MASS, and so were formed into a lookup database that MASS could link to its pointing budget.

Simulation Results

MASS achieved functional completeness in mid-October 2022. Since then, it has been tested against SSRL's two primary CubeSat missions to error-check the software and to perform preliminary validations of these missions' operational concepts. The results of this testing have been highly encouraging for MASS's applicability. This testing has both validated many parts of MASS and each CubeSat mission and revealed a large variety of potential improvements for all the above.

Mission Concept Validation

One of the primary concerns around the MOCI satellite during initial development was the power-positivity of its systems. With a number of power-hungry systems, the possibility of causing excess drain to the battery and leading to an emergency safe mode trip is very real, especially during intense scheduling. MASS's mode determination criteria were able to overcome this issue and efficiently schedule an extreme number of mission-critical modes without unduly draining batteries during the course of the simulation. For the MOCI team this was very affirming and confirmed MOCI's potential by predicting that MOCI could hypothetically complete all mission success goals within a single week, though MOCI will not be subjected to this intense of a schedule post-launch to allow for post-launch verification. Validation of power-positivity was later repeated for MEMESat-1. Figure 6 shows a cumulative log of the primary three modes over time, which illustrate MASS's ability to minimize unused downtime and avoid any wasted time via efficient automatic scheduling.

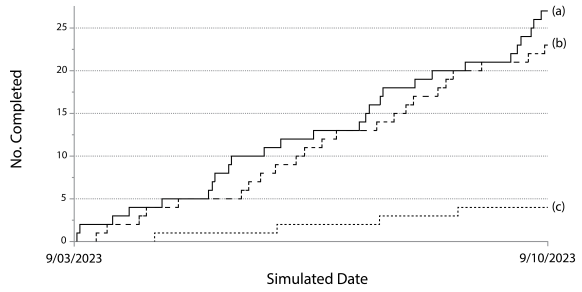

Figure 6: Cumulative number of sequences completed over time for the data (a) gathering mode, (b) processing mode, and (c) downlink mode

MASS did also reveal necessary improvements in the operational concept of MOCI. Chief amongst these was extensive fine-tuning of mode determination criteria. Various parameters such as the allowable minimum state-of-charge to begin certain modes, or the minimum allowed elevation to contact the ground station, were tweaked a number of times in order to achieve a productive scheduling balance. To use the ground station as example, if the allowable elevation is set too low, MASS will preclude necessary modes such as data processing in favor of low-quality passes, therefore degrading scheduling efficiency. Additionally, MASS revealed that MOCI's link budget struggled to keep up with the massive data products that MOCI routinely generates, and this can be seen in the lag in slope of the mode (c) in Figure 6. This data led to changes in MOCI's data budgeting that were able to rectify this issue.

Potential Improvements & Future Research

MASS has a number of areas of opportunity for upgrades and improvement. As previously mentioned, some calculations such as the power generation equations are simplistic and unrealistic, implemented more for functionality than accuracy. These can either be improved within MASS's functions, or alternatively they can be linked to coupled scripts in languages like MATLAB or python. Improvements to streamline the user interface and make the program simpler for a new user to learn and utilize will also be implemented. These will include user prompts to set CubeSat properties such as battery and data capacity, or antenna capabilities. Another unique improvement will be to include weather forecasting in MASS's calculations for ground station contact scheduling, as in works such as the 2020 paper Wang et al.[10] One interesting potential improvement would be to create an abstracted version of MASS with completely generic budget and scheduling components, which may be helpful for users who need to extensively modify MASS to accommodate unique and novel CubeSat missions.

A final intended overhaul for MASS is to add a form of data assimilation. Data assimilation is the process of intaking satellite-collected telemetry and nudging a time-dependent simulation, in this case MASS, to gather more accurate predictions. Data assimilation was pioneered in the Apollo lunar missions, and is most famously used in numerical weather prediction. A simplified method can be applied to MASS, in which acquired telemetry from a CubeSat can be initialized into the simulation to provide an accurate starting point for the CubeSat's current state. This would eliminate accumulative predictive error during each ground station contact.

The essentials of this method can be seen in Figure 7. Overall, this would allow MASS to become an extremely accurate and well-coupled predictive model for arbitrary ambitious CubeSat missions.

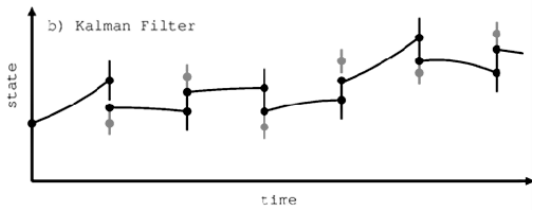

Figure 7: Shows the overall process of Kalman filtering, which is similar to MASS's data assimilation procedure, albeit adding interpolation and model parameter forcing.

MASS INTEGRATION

Though integration of the MASS system into satellite operations requires some work on the satellite side, the system's configurability significantly lessens workload of the end user. When designing a satellite's flight software (FSW) to integrate MASS, the goal is not to provide full autonomy to the satellite, but rather to automate the parts of the process necessary for the mission to more easily operate outside of the direct control of ground station operators. An integrated FSW should also provide a direct throughput of telemetry for MASS to operate on. This goal can be seen in the concepts of both the MOCI and MEMESat-1 satellites (which will incorporate MASS into its mission operations through the scheduling of their operational modes). Due to MOCI's higher task complexity, it will be the primary model as an example of integration. After the MOCI mission's FSW design is introduced, we will then show how the design of MASS and MOCI will work together to create a complete functional system that provides support to mission operators.

Introduction to MOCI

The MOCI mission's primary goal is to acquire imagery of the Earth's surface from LEO and perform real time Structure from Motion (SfM) at a landscape scale using custom algorithms and commercial off-the-shelf, high performance computational units. Efficient data compression, feature detection, feature matching, and SfM processing techniques of space-based imagery will be performed on board the spacecraft.[11] In order to achieve this goal, it is necessary for the payload to take images of the Earth's surface when it is outside the immediate control of ground station operators. MASS, as a centralized, automated scheduling system will help the satellite complete tasks at specific points in time.

These operational modes are designed with the methodology of a Finite State Machine in mind. As mentioned previously, MASS models this, then works to determining what modes need to run when, simplifying the flight software integration. Ground station controllers may command the satellite to assume five separate nominal operational modes defined as follows: Cruise Mode, a power positive idle state; Power Generation Mode, a power positive idle state used when maximum charging capabilities are required; Scan Mode, a target tracking mode used to collect images of the Earth's surface; Data Processing Mode, the primary computational mode to process collected imagery; and Data Downlink Mode, the science communication mode for transmitting processed data over S-Band radio. In addition to these nominal modes of operation, there is also Safe Mode which can be used by the satellite to autonomously identify and prevent non-nominal operations within the software or hardware. This mode can also be manually entered by ground controllers if such action is deemed necessary.

Introduction to Operational Rules

For the integrity of the collected science data, MOCI operational rules are designed around the idea that tasks must not be interrupted, either by another task or by a preventable non-nominal event such as low battery voltage. The operational rules are split across both MASS and MOCI because of the nature of their conditions. The onboard flight software has limited data to work with, primarily health and storage telemetry. For any navigational prediction, a simulation must be run. Therefore, the flight software requires an extensive but simple set of configuration data set by ground controllers in order to know what to do and when to do it. This ultimately means that a majority of the rules are implemented within MASS, rather than onboard MOCI, in order to assist ground station controllers in the scheduling process.

There are generally four initial conditions that will trigger a transition between modes: a manual command from the ground station, a scheduled command from the scheduler's queue, the completion of a task, and an automated transition to Safe Mode because of an anomaly. However, these conditions are not applicable in all cases. For example, a scheduled transition cannot be accomplished from Safe

Mode. A summary of these rules are given in Table 3, where each entry describes which conditions can be used to transition from the current mode, listed in the left column, to another mode, defined across the top. It is understood that a mode can transition to itself without error but may be initially blocked by a task that is being completed. Beyond these initial conditions, the flight software will also perform checks to confirm the health of the satellite before beginning a task. These operational rules will be outlined for each individual mode in following sections.

Table 3: A summary of mode transition conditions, including: (1) manual command, (2) scheduled command, (3) task completion, and (4) anomaly

	Safe	Cruise	Scan	DP	DD
Safe	1 2 4	1	0	0	0
Cruise	1 2 4	1 2	1 2	1 2	1 2
Scan	1 2 4	3	3	0	0
Data Proc	1 2 4	3	0	3	0
Data Down	1 2 4	3	0	0	3

Cruise Mode Operational Rules

Cruise mode has the simplest operational rules. Since the satellite is in an idle state in this mode, this mode is always the current state when no task needs to be done on the satellite. As such, the entry criteria for cruise mode is simply that all current scheduled tasks are complete. As shown in Table 3, Scan, Data Processing, and Data Downlink modes can only transition to themselves or cruise when they are done with their tasks. Cruise mode can also be manually entered from safe mode whenever troubleshooting is complete.

In order to exit cruise mode, there must be a scheduled or manual entry to another mode as shown in Figure 8. Any criteria to enter that mode will be checked based on which mode is being entered.

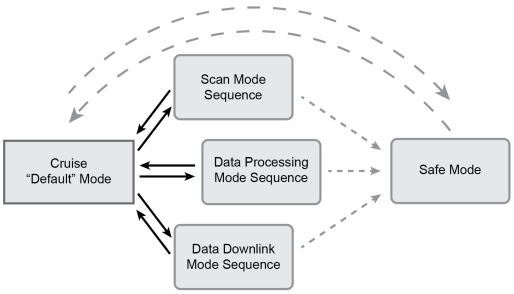

Figure 8: Cruise mode valid transitions.

Scan Mode Operational Rules

Scan mode is the primary science data collection mode, and therefore has strict rules on when and how it should operate. In order to enter a scan, the following navigational and mission-based checks shown in Table 4. must be made.

Table 4: Scan Mode Rules

A scan of a ground target should not be considered within 50 minutes of a ground station pass in order to maximize available antenna time.
A scan should only be scheduled to start at five minutes prior to a pass directly over a target.
Targets should be presented to the payload in the daytime and at an optimal angle compared to other possible targets in a valid scan period.
A target should be at the highest priority for the available scan period. This priority list is set by the mission operators and can change based on mission needs.
Battery state-of-charge should be over 75%. This is also verified by FSW.
There should be available storage onboard for collected data. This is also verified by FSW.

The method for target priority determination by mission operators has additional criteria that bear further exploration. As MOCI uses stereoscopic SfM, its primary targets are mountains with large terrain relief. Therefore, high plateaus with low elevation variability are poor targets despite their high altitude. As such, for MOCI a list of targets has been compiled on the basis of topographic prominence: the height of a summit relative to the lowest contour encircling it but not containing a higher summit. Mountains such as Mauna Kea have higher prominence than other, higher summits like K2, as they are isolated, and therefore more prominent. If two valid targets overlap, Scan Mode will prioritize the target with higher topographic prominence.

Once all images are taken, the FSW will confirm that the data has been stored properly and that the payload is powered off before transitioning back to Cruise Mode.

Data Processing Operational Rules

Data processing mode is the most complex to administrate due to the considerations necessary to manage its high power draw. There are several conditions to enter data processing mode that can help with this challenge along with the general rules of operation as shown in Table 5. The conditions required to subsequently exit data processing mode and enter cruise are shown in Table 6.

Table 5: Data Processing Mode Entry Rules

A data processing period should be planned to start when the satellite first enters the sunlight after being in eclipse.
A processing period should not be considered within 75 minutes of a ground station pass in order to maximize available antenna time.
A processing period should not be considered within 75 minutes of the start of a priority target pass.
There must be images to process stored onboard. This is also verified by FSW.
Battery state-of-charge should be over 75%. This is also verified by FSW.

Table 6: Data Processing Mode Exit Rules

Data has been completely processed by either pipeline, compressed, packetized, and properly stored both on the payload computer and onboard computer.
There is a manual or automated command to end the processing session for any reason. In this case, the processing state is saved to be resumed later.
The processing takes longer than 60 minutes. In this case, the processing state is again saved for later completion.

Data Downlink Operational Rules

Due to the limited infrastructure of the ground station network, antenna time is the highest priority during mission operations. As shown in Tables 4 and 5, this mode is prioritized over other modes onboard the satellite due to it's limited availability compared to other periods. In order to enter data downlink mode, three conditions must exist as shown in Table 7.

Table 7: Data Downlink Mode Rules

The satellite is passing the ground station with an elevation above 20 degrees.
There is stored data to downlink.
Battery state-of-charge is sufficient to complete the downlink. This is also verified by FSW.

After successful downlink, the goal is for the ground station to have sufficient pass time to send commands or schedules to MOCI. However, if the downlink is predicted to take the entire pass period, then it should be scheduled with another valid pass directly after it to allow adequate science data download and uplink time.

MASS to MOCI Pipeline

As shown in the prior sections, the MASS system has a lot to consider when scheduling tasks for MOCI. The design of MASS as discussed in the first section is meant to improve the efficiency of these considerations over a ground station operator planning a complex schedule by hand. In order to communicate these decisions to the satellite, the onboard scheduler and mode management system was designed with a simple configuration system. Each mode is assigned a number along with a set of parameters that can be set to define when a scheduled task is to be executed. These parameters differ for each mode but consider critical information such as target latitude and longitude, which camera is to be used, the target's ID number, and much more. This configuration system allows MASS to output a procedural sequence of modes based off its simulations. Once this plan is produced, ground station operators can verify and approve the plan, and once approved by a ground station operator, the configurations are packetized and prepared for transmission to MOCI.

Once MOCI makes a pass, this plan can be uploaded to MOCI while telemetry data is downloaded from MOCI. Once onboard, the scheduler begins timed interrupts for the beginning of each task. Once these interrupts execute, all the needed checks occur for the desired mode transition and the task begins. The telemetry that is received from MOCI is also of utmost importance for this process. Due to the fact the space environment is always changing, the downloaded telemetry data can be fed back into MASS in order to run the next batch of simulations with the most up to date information on the satellite. Any changes in position, battery voltage, storage space, etc. can then be more accurately simulated by MASS to ensure the next schedule fits the constantly changing needs of the satellite. This control flow is outlined in Figure 9.

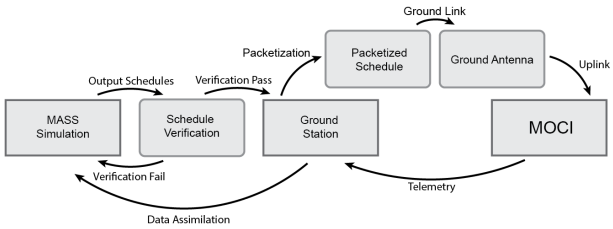

Figure 9: The flow of scheduled tasks and telemetry data to and from MOCI and MASS.

Mass Verification

Due to the importance of the MASS pipeline to mission operations applications for MOCI, testing is of utmost importance if SSRL is to fully rely on the system. As the goal of MASS is to assist in post-launch mission operations, testing the system before

that point is difficult to accomplish. Therefore, while post-launch verification will be the primary test of the system, pre-launch testing will also play an important part in MASS evaluation. This section will outline the changes made to common tests in order to integrate MASS and will specify the tests that can be used to evaluate MASS's success in assisting mission operations post-launch.

Pre-Launch Verification

Thoroughly testing MASS on the ground proves difficult for a couple reasons. First, the environment on earth is much easier on the flight hardware of the satellite. As such, our budgeted data is more reliable on earth compared to what we might expect in the space environment. Second, the satellite is not physically orbiting earth, so any variations that might happen in orbit are hard to simulate. MASS simulates a number of different factors that affect orbital motion such as Earth's oblateness, the influence of the Moon and the Sun, aerodynamic lift, and radiation pressure, making it very accurate. Nevertheless, cumulative simulation errors or unforeseen physical phenomena cannot be conclusively removed from ground testing. This ultimately means the closest we can get to running MASS as we would post-launch is by extensively testing MASS based on scheduling scenarios and Day-in-the-Life (DITL) testing.

Scheduler Testing

The Mission Operations team tests the capabilities of MASS by repeatedly running sample scenarios through the simulation. One of the first tests is valid mode scheduling. Operators manipulate the mode timings in MASS in order to verify whether the complex mode determination system has properly generated the sequence of modes expected by the team. MASS's MOCI implementation is required to abide by the previously defined MOCI Flight Rules (MFR), including such safety precautions as prohibiting battery discharge to lower than 20% state-of-charge or powering down the payload computer when transiting over the South Atlantic Anomaly.[12] When testing MASS, operators determine whether MASS is following these principles, which is expedited by MASS's ability to change the output variables to any number of different diagnostic states. This allows for easy debugging of the code when this output does not match a success state. So far MASS has been confirmed to present proper and error-free mode scheduling as of 2023.

MASS's more complex functions, which do not necessarily appear easily through the scheduling process, must also be verified. As an example, data budgeting is highly complex and mutable based on what the FSW Team determines to be most effective for managing and compressing mission-critical data products. This is evidenced by the MASS results demonstrating a need for changes to the link budget software, de-prioritizing large intermediate data products in favor of the smaller finalized height maps that MOCI produces, seen in Figure 10. These necessary changes require coordination between the FSW and Mission Operations Teams to ascertain the proper success states for MASS mode simulations. Verifying these more complex budgets within MASS is completed through more involved testing, namely alongside DITL testing.

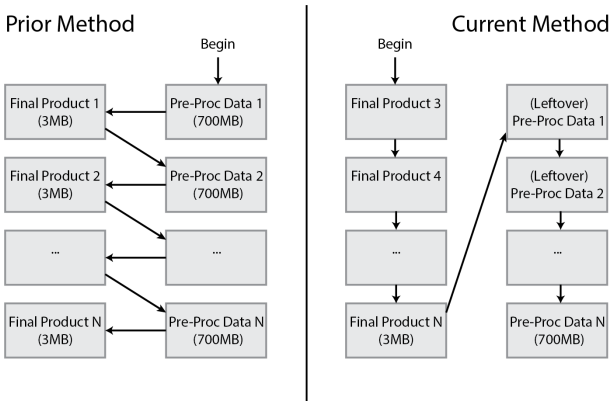

Figure 10: MOCI's data downlink priority scheme before and after MASS determined that the large intermediate data packets used to double check the final products, were hindering final product delivery due to link budget constraints.

Mode Simulation Testing

DITL testing is an accepted and common method of flight readiness testing, and every space bound satellite has gone through some iteration of this testing. However, the incorporation of MASS changes our testing by removing the need to hand-write schedules. MOCI's DITL testing is accomplished using a "flatsat" representation of its electronics stack to test the software implementation, whose main components are displayed in Figure 11. DITL tests are run over a 24-hour period and are run with the same operating procedure that the satellite would use in space.

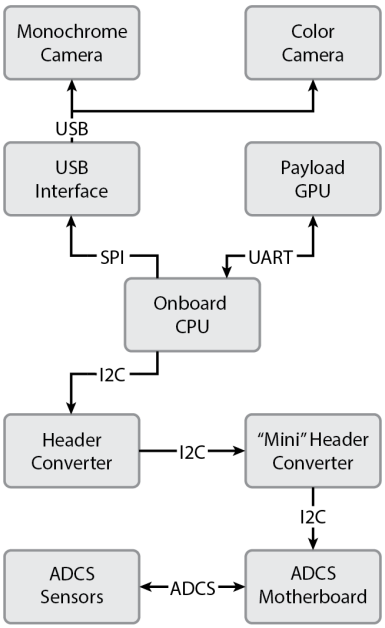

Figure 11: A diagram of the main components of the MOCI "Flatsat" testing rig, and the types of connections that each component uses to communicate.[13]

At the beginning of a DITL test, MASS will produce a simulated schedule for the 24-hour duration of the test. This schedule should be approved by an operator and uploaded to the engineering satellite just as it will be done in post-launch operations. The DITL test will then be observed for the 24-hour period, as there should be little to no operator intervention once the schedule is uploaded. Ground station pass times are simulated into the test as well, meaning operators may only intervene during a pass time. During testing we often relax this rule for debugging in order to save time until final system tests are ready, but any nominal operations, such as a new schedule upload or data downlink, follow this restriction.

For verification of MASS's ability to properly simulate modes and budgets, MASS's power log values are analyzed to confirm that the correct power draws are being applied during each mode transition, and to ensure power-positivity. When DITL tests are performed, the modes are performed on the Flatsat as they would be in orbit, and the real budget values ranging from data management to battery drain are subsequently obtained. These logs can be cross-referenced with MASS simulations to determine whether MASS is accurately representing the dynamics of these modes, or potentially, whether there is a hidden error during the DITL causing the Flatsat to perform differently than intended. When the Flatsat proves able to complete any valid schedule produced by MASS regardless of intensity, this will affirm MASS is capable of generating safe schedules that abide by the MFRs.

Lifetime Estimation

One primary area of testing that will be important to MOCI and MASS verification is lifetime estimation. Due to a recent ruling by the FCC in 2022, all low-Earth orbit satellite operators must deorbit within 5 years.[14] Orbital lifetime simulations have been performed for decades, and studies such as King-Hele have produced useful lookup tables like the one shown in Figure 12.[15] The necessity of this simulation for FCC compliance represents an excellent opportunity for confirming MASS simulation accuracy. Given that the equations and general results for orbital lifetimes are known, we can cross-reference simulations performed in MASS, such as Figure 13, to known results and check for any discrepancies. MOCI has been allowed a range of possible orbits between 45-110 degrees of inclination and between 350-500 kilometers in altitude, and lifetime estimations over this range are in progress through MASS as of June 2023.

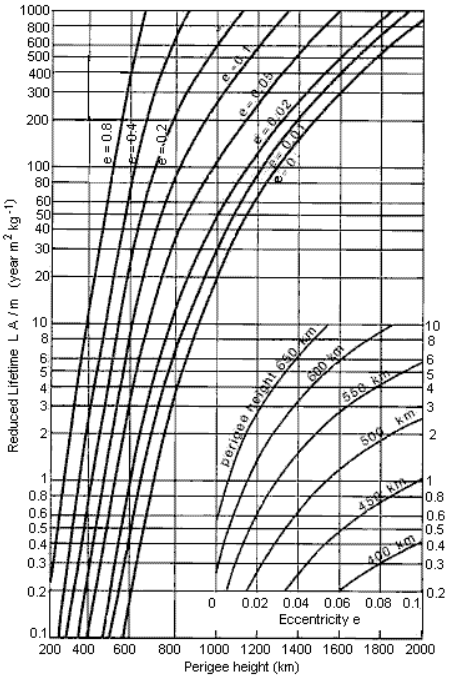

Figure 12: A series of orbital lifetime curves for different orbital parameters including altitude and eccentricity, as determined by King-Hele (1987).[15]

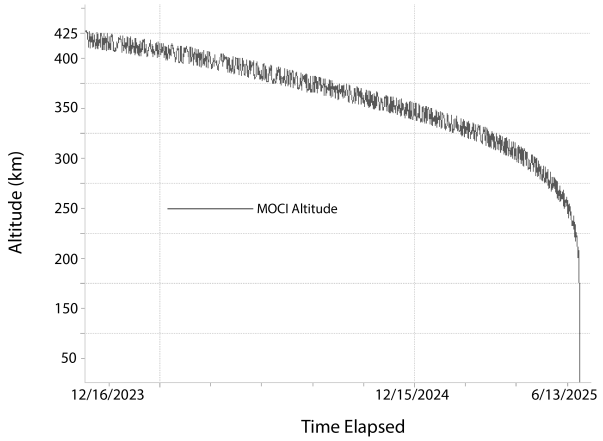

Figure 13: A MASS simulation of MOCI's orbital decay beginning at an altitude of 425 kilometers, resulting in a lifetime of approximately two years.

Post-Launch Verification

Once MOCI is onboard the launch vehicle, the Mission Operations team officially shifts over to the satellite operations role. One of the first responsibilities of the Mission Operations team is to prepare for operators in the Mission Operations Station for the satellite's first radio signal. During this initial period, the Mission Operations team will begin operating the ground station to establish communications with MOCI as soon as the satellite deploys, and then begin tracking. Once communication is established and MOCI is deemed nominal, operators will begin generating the first set of operational schedules using MASS. It is during this period that definitive testing can be done. Once MOCI is able to achieve nominal status, its telemetry can be used to evaluate MASS conclusively.

Operational schedules for MOCI will contain at most a 7-day nominal operational schedule. This will coincide with a 14-day contingency operational schedule in order to enable MOCI to continue acquiring data despite any possible ground station anomalies, given that MOCI's onboard clock can execute time-based mode transitions automatically. Due to orbital shifting and potentially outdated TLE files, the duration of each schedule should ideally be kept at a minimum, and new schedules should be uplinked as frequently as possible. The frequency of schedule updates will be dependent on how often the SSRL ground station can communicate with MOCI and uplink a schedule and will also be dependent on MOCI's undetermined orbit. MASS has determined that on average, MOCI will pass over SSRL's UGA ground station once per day. The MASS pipeline factors in data acquisition and storage limits, mode transitions and duration, as well as slewing and will incorporate up-to-date data files for MOCI via downlinked telemetry whenever possible.

MASS Pipeline Testing

Once post-launch, MASS will begin the testing of real-time telemetry data being read into it via data downlinked from the most recent of MOCI's passes. During this phase of the satellite's lifetime, three important pieces of data must be verified by MASS in order to deem the verification of MASS as a success.

First, MASS must be able to manage an automated process of data assimilation from downlinked telemetry, execution of schedule and event prediction, and delivery of full and valid schedules to mission operators without intervention at any point during the process. Verification of this capability follows fairly naturally, as delivery of the product will indicate functionality of the pipeline. Second, MASS must properly abide by all current or future MFRs and mode determination criteria as detailed in the MASS integration section above. To verify that these rules are being accurately accounted for by MASS, mission operators can use MASS's extensive logging capabilities to output debugging variables that can record any instances of error. Lastly, MASS must accurately simulate each of the essential budgets that influence its scheduling: power, data, and link. This is perhaps the most important area to verify for MASS to become a reliable part of the SSRL workflow, as if MASS cannot reliably determine crucial variables such as battery state-of-charge or data capacity, the resulting schedules could become dangerous if utilized. To verify that the necessary key orbital parameters are being accurately generated by MASS, mission operators will use MOCI's telemetry to fact check their MASS generated schedule. This cross-referencing will procedurally check that MASS is efficiently generating the most accurate daily mode schedules possible. In summary, proving MASS's capability to quickly generate schedules will allow MOCI and MEMESat-1 mission operators to save hours of a mission operator's time and standardize the data that MASS and these missions should input and output to produce accurate results.

CONCLUSION

As the aerospace industry moves towards commercialization and autonomy in satellites, demand for automation solutions to scheduling are increas-

ing. Simultaneously, universities like the University of Georgia are increasingly beginning to push the boundaries of novel CubeSat missions, and the Small Satellite Research Lab hoping to continue this trend via the MOCI and MEMESat-1 missions. Many university aerospace teams have limited resources that preclude full and complex autonomy, and look to hybrid approaches that expedite scheduling without sacrificing careful real-time human oversight. Here, we have presented our preliminary results and procedures for integrating a predictive scheduling software, the Multi-aspect Automated Satellite Scheduler. This novel FreeFlyer-based program is intended to be usable by any CubeSat mission and fulfill the needs for accessible hybrid automation in the accelerating aerospace industry.

MASS validation is difficult to accomplish prelaunch, but a number of tests have been conducted to verify components such as pass prediction, budget simulation, and mode scheduling via cross-referencing accurate values. These tests have confirmed MASS to be internally consistent, accurate, and error-free. Upon MOCI's launch, currently expected in 2024, and the subsequent launch of MEMESat-1, we expect to confirm MASS's ability produce large and highly efficient schedules, and subsequently reduce human involvement in scheduling.

Future research will include improvements to the MASS software, such as further integration with the SSRL ground station, and increasing the fidelity of MASS's simulation to improve accuracy. Exhaustive completion of additional functionalities for MASS will include data assimilation components to intake telemetry from CubeSats, and to further improve the end user experience when implementing MASS in operational pipelines.

Ultimately, MASS represents a large improvement in automation for SSRL's own scheduling and budgeting procedures, and as we continue to improve MASS, we hope it will be able to become an invaluable tool to other teams in need of solutions to CubeSat scheduling.

Acknowledgments

The authors would like to thank the Air Force Research Laboratory's University Nanosatellite Program for giving us the opportunity to work on MOCI through their funding and support. We would also like to thank our principal investigator Dr. Deepak Mishra for supporting the lab and our research and Sydney Whilden, our lab manager, for her support and assistance throughout our research efforts. Finally, we would like to thank A.i. Solutions for providing us with their software for free to develop this system.

References

[1] Romain Grasset-Bourdel, A Flipo, and G Verfaillie. Planning and replanning for a constellation of agile earth observation satellites. *ICAPS 2011*, page 29, 2011.

[2] Xiaogeng Chu, Yuning Chen, and Yuejin Tan. An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. *Advances in Space Research*, 60(9):2077–2090, 2017.

[3] Seung woo Baek, Sun mi Han, Kyeum rae Cho, Dae woo Lee, Jang sik Yang, Peter M. Bainum, and Hae dong Kim. Development of a scheduling algorithm and gui for autonomous satellite missions. *Acta Astronautica*, 68(7):1396–1402, 2011.

[4] Maria Theresia Wörle and Christoph Lenzen. Ground assisted onboard planning autonomy with vamos. In *International Workshop for Planning and Scheduling in Space*, March 2013.

[5] Xiaoyu Chen, Gerhard Reinelt, Guangming Dai, and Andreas Spitz. A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research*, 275(2):694–707, 2019.

[6] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal*, 15(3):3881–3892, 2020.

[7] Wei-Cheng Lin, Da-Yin Liao, Chung-Yang Liu, and Yong-Yao Lee. Daily imaging scheduling of an earth observation satellite. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(2):213–223, 2005.

[8] Jian Wu, Yuning Chen, Yongming He, Lining Xing, and Yangrui Hu. Survey on autonomous task scheduling technology for earth observation satellites. *Journal of Systems Engineering and Electronics*, 33(6):1176–1189, 2022.

[9] A.i. solutions Inc. *Spacecraft Propagators; Propagator Summary Table*, May 2023. Accessed: 2023-06-06.

[10] Xinwei Wang, Guopeng Song, Roel Leus, and Chao Han. Robust earth observation satellite scheduling with uncertainty of cloud coverage. *IEEE Transactions on Aerospace and Electronic Systems*, 56(3):2450–2461, 2020.

[11] Caleb Adams, Nicholas Neal, and David Cotten. *MOCI Requirements Verification Matrix*. Internal Document, 2022.

[12] Bjorn Leicher, Paige Copenhaver, Graham Grable, Adam King, Caleb Adams, Sydney Whilden, and Jackson Parker. *MOCI Concept of Operations*. Internal Document, April 2019.

[13] Cameron Bonesteel, Evan Tichenor, Eric Miller, and Andres Rodriguez. Co-operating systems: A technical overview of multiple onboard operating systems. *Small Satellite Conference*, August 2022.

[14] Federal Communications Commission. Space innovation ib docket no. 22-271 mitigation of orbital debris in the new space age ib docket no. 18-313, September 2022.

[15] D.G. King-Hele and D.M.C. Walker. The effect of air drag on satellite orbits: Advances in 1687 and 1987. *Vistas in Astronomy*, 30:269–289, 1987.