



**AIAA – RS4-2006-7001**

# **AUTONOMOUS OPERATIONS FOR RESPONSIVE SPACECRAFT**

Jackie Reilly, Systems Engineer,  
a.i. solutions, Inc., Lanham, MD

Terrance Yee, Systems Engineering Manager,  
Microsat Systems, Inc., Littleton, CO



**4th Responsive Space Conference**

April 24–27, 2006

Los Angeles, CA

## AUTONOMOUS OPERATIONS FOR RESPONSIVE SPACECRAFT

By Jackie Reilly, Systems Engineer, a.i. solutions, Inc, Lanham, MD  
and Terrance Yee, Systems Engineering Manager, MicroSat Systems, Inc., Littleton, CO

### ABSTRACT

MicroSat Systems, Inc. (MSI) is currently supporting the Air Force Research Laboratory (AFRL) on several responsive space programs demonstrating tactically useful small satellites with autonomous operations. Contrary to previous autonomy efforts, these missions have autonomous on-board software which is specifically designed to allow end users, with no satellite expertise, to task the vehicle directly. This capability puts the Warfighter in direct control of a spacecraft, changing several aspects of the original paradigm throughout the mission.

The use of autonomy tools and modular software elements change the paradigm for traditional spacecraft operations. The change in the operational concept creates implications for overall project development timelines and for the end user in the field. Autonomous operations must be accounted for at the beginning of the requirements definition phase. While it is possible to upgrade existing systems after they are already designed, it is much more cost effective to initially design with the non-expert end user in mind. This allows the appropriate selection of command and telemetry architecture to accommodate both the traditional expert end user and a specialized interface for the non-expert end user.

Copyright © 2006 by Jackie Reilly, a.i. solutions, Inc and Terrance Yee, Microsat Systems Inc. Published by AIAA 4<sup>th</sup> Responsive Space Conference 2006 with permission.

Designing the non-expert interface to be simple and minimalist from the beginning has profound impacts on the structure of the entire autonomous software.

There is a major impact on spacecraft integration and testing throughout the development cycle and during on-orbit commissioning. There is significant synergy gained from coordinating the automation software (that performs ground testing), on-board testing (to verify state of health), and commissioning the spacecraft on-orbit. By careful design of these capabilities, it is possible to not only save time but also increase the degree to which the ground team can follow the “test like you fly” principle. The net impact of these changes is the following: shortens the time needed to deliver a working product to the end user, brings the end user’s concerns closer to the design team, changes the focus of the spacecraft utility design from a strategic asset to one which has short term tactical significance, and places extremely powerful space assets in the hands of ground forces within minutes of their request. This places a unique set of requirements on the software to be as easy to use as possible while also ensuring the safe operation of the spacecraft. Pairing the expertise of ground support and current autonomous ground system software with the expertise of spacecraft developers and current on-board software will help to design the on-board autonomous software that accomplishes this task.

### INTRODUCTION

MSI and a.i. solutions have worked on several missions at the forefront of the application of increasing autonomy for Low Earth Orbit (LEO) spacecraft. These include the TacSat series, TechSat-21, Earth Observation System (EOS) and the Solar Dynamics Observatory (SDO).

a.i. solutions, Inc. and their COTS (commercial-off-the-shelf) software product, FreeFlyer®, have successfully supported analysis and operations of ten years of NASA, NOAA, Military and Department of Defense missions and programs. The below missions are supported with FreeFlyer® and engineering services that included pre-launch analysis, launch support, and on-orbit analysis:

YEAR	MISSION
1997	TRMM
1999	EOS Terra
1999	Landsat-7
2000	EO-1/SAC-C/Munin
2002	EOS Aqua
2004	EOS Aura
2004	GPS IIR-11, IIR-12, IIR-13
2005	GPS IIR-14
2006	GOES-N
2006	CALIPSO & CloudSat
2006	ST-5
2006	GPS IIR-M2
2006	THEMIS
2008	SDO
2010	GPM
2011	Crew Exploration Vehicle
2013	James Webb Space Telescope
2013	MMS

Table 1: Selection of Missions supported by a.i. solutions, Inc. and FreeFlyer®

Of particular interest to the content of this paper are the TRMM, EO-1, and GPS missions where FreeFlyer® was used in an autonomous and semi-autonomous

environment on the ground, and in the case of EO-1, on-board.

These missions and others currently in development have begun a trend toward increasing capabilities of the on-board flight software to autonomously operate the spacecraft. These capabilities are not mere conveniences for the traditional mission controller at the central ground station, but powerful tools that can change key ways in which small satellites are used and developed. The recent advances in autonomy software are allowing small, responsive spacecraft to be developed for Earth orbiting missions which can find, diagnose, and repair faults, calculate propulsive maneuvers and execute them to achieve a specific mission objective. Software can also commission the spacecraft once it reaches orbit, plan and execute payload operations including on-board data analysis to generate calibrated, rectified data end products, and schedule the full range of spacecraft activities of varying priority over the course of weeks. By replacing the functionality of skilled operators, engineering experts, schedulers, and data analysts, advanced software can allow missions to be run with just a mission director for the vast majority of its lifetime.

### OPERATIONS

In the past, in order to solve program requirements and design a spacecraft orbit, flight dynamics engineers (FDE) used many tools to develop a mission plan for each unique set of requirements. After many hours, the FDE had a plan and any change to this plan often took substantial effort to debug and test the updated system. COTS analysis software products, like FreeFlyer®, enable the engineer to solve more complex problems in a shorter period of time, by allowing the engineer to repeatedly run trade study analyses without software reconfiguration.

FreeFlyer® needs to be configured one time in order to generate the desired output and can be run, autonomously, without changes, for the lifetime of each responsive space mission. Once deployed, the same software can be used for a series of responsive spacecraft missions.

Three missions that help to show the impact of using a COTS tool for mission planning (pre-launch) and operational product generation (pre and post launch and on-orbit) are EO-1, TRMM, and the GPS system.

NASA's Earth Orbiter-1 (EO-1) used the FreeFlyer® precursor, AutoCon™, to perform maneuver planning, orbital analysis, and product generation automatically on the ground and on-board. EO-1, with AutoCon™ was able to utilize a user-developed algorithm to plan and execute maneuvers for formation flying with the LandSat-7 spacecraft. On-board, AutoCon™ used its algorithms to automatically plan future maneuvers as well as calibrate executed maneuvers to provide accurate orbit states. On the ground, FreeFlyer®'s fuzzy logic capabilities enabled ground system analysts to solve multiple conflicting orbit constraints autonomously. For example, fuzzy logic determined the best times to perform maneuvers, incorporating the ground analyst's schedule. FreeFlyer® was used to streamline the ground based maneuver planning system used for these spacecraft as well as the subsequent Earth Observing System (EOS) spacecraft, Terra, Aqua, and Aura.

To assist in the coordination of the EOS Constellation, which includes the EOS missions mentioned above and eight other missions, EO-1, LandSat-7, SAC-C, Parasol, and future missions CloudSat, Parasol, CALIPSO, Glory and OCO, a.i. solutions, with NASA Goddard, developed the Constellation Coordination System (CCS). CCS helps coordinate the entire constellation

of spacecraft over the internet. This system allows the electronic exchange of data between the constellation spacecraft and delivers information to mission managers, scientists, and researchers. This service also gives access to FreeFlyer® visualization, scheduling tools, and mission products (eg: output ephemeris files, contact times to ground stations, etc.). Access to the CCS website allows analysts and mission managers to exchange and view data and conduct analysis when necessary.

The Tropical Rainfall Measuring Mission (TRMM) used a.i. solution's product, AutoFDS, with FreeFlyer® to reduce the cost of the daily flight dynamics (FD) operations of the NASA Goddard mission. AutoFDS uses a single spacecraft ephemeris product, provided by the NASA Flight Dynamics (ground) Facility (FDF), to automatically create all FD products needed to control the daily operations of TRMM. Prior to AutoFDS, the ground Flight Operations Team (FOT) had to manually generate the products individually. AutoFDS reduced the need to maintain legacy software by replacing over 13 different applications with one piece of software. Now TRMM FD operates in an autonomous environment that verge on 'lights-out'.

In the GPS example, FreeFlyer® interacts with an external interface designed by the end user. Using TCP/IP sockets, a command is sent from the external interface to spawn an instance of the FreeFlyer® application and a "wizard list" allows the end user to run any number of analysis and mission planning scenarios. The scenarios listed on the wizard list are tailored to solve specific tasks for the GPS project and are used for mission planning and to generate mission specific products. FreeFlyer® interacts with Matlab®, as needed, using an API and displays native user interfaces within the FreeFlyer® environment to easily enable the non-expert end user to

interact with expert aerospace mission applications. The native user interfaces for GPS were designed by a.i. solutions with “use case” specifications for the non-expert end user. Dynamic inputs caused by changes in the orbit and attitude states of the spacecraft are queried from a database, and all spacecraft orbit, attitude, and maneuver properties are fed to the database as instructed by the user. This makes it easy for the user to archive all spacecraft states and parameters to allow for future testing and more detailed analyses.

The above three cases highlight that fact that with more autonomy in a system, the more time and cost effective the system can be. By utilizing tested applications, the expert analyst time is not consumed by maintenance and training new users on legacy systems. The end user can also simply learn a system that can easily be configured for any program, one time. This reduces the cost of training each individual flight operations team repeatedly.

### **DEVELOPING AUTONOMOUS SOFTWARE**

Autonomous software must be designed for the non-expert end user. Use case scenarios enable the creation of a simplified command and telemetry user interface, similar to the architecture used in the above GPS mission, to act as a layer to the more detailed expert system traditionally used. Color-coded indicators (eg: Red/Green indicator representing complex state of health of whole system) with simplified command sets are one example of ease of use available to the end user. Built-in features can check or validate commands on the ground and on-board. Quality Assurance (QA) checks include vehicle state of health, thermal limits, power states, and attitude control margins. Future additions can include, advanced long range planning to schedule various operations automatically. Finally, expert users have the

ability to lock or give access to specific tactical non-expert end users.

Telemetry products often require a separate on-board processor or data product processing software to replicate ground expert data evaluation and generation of finished products. With on-board autonomous software, end products, often maps or calibrated geolocated images can be created on-board and relayed down to the end user. This replaces the need for two separate software products. If the on-board analyses and command product calculates the same data as the ground product, testing will be more accurate and less time consuming. These two things ensure that the final software product for the autonomous spacecraft is the most cost effective.

Using the various constraints inflicted on the spacecraft, the ground and on-board software can be configured to run different analysis cases, including those utilizing fuzzy logic. The cases can determine which spacecraft is the most useful for the situation based on visibility of a ground sight, battery power, line of sight, available telemetry or any other variable of concern. After initial set-up, the on-board software would run autonomously to solve for the most desirable conditions. Once solved, the spacecraft would send a “GO/NO-GO” command to the ground and to the other spacecraft of interest. The ground software, if needed, would allow simple interaction with the non-expert end user. Depending on the set-up, the spacecraft can automatically act on a “GO” or it could wait for a command from the ground.

Using FreeFlyer®, a.i. solutions created a “Sensor to Shooter” example case. In this scenario, a spacecraft in a constellation of five, receives a command from the ground to take an image of a ground location. The initial spacecraft autonomously sends the

command to the other spacecraft in the constellation, and the spacecraft that has the most desirable view, takes the image. Once the image is taken, the spacecraft relays the image to the rest of the constellation, and the image is sent back to the ground. This example shows that little ground interaction is actually needed while using a constellation with on-board software. The spacecraft are able to communicate with each other leaving the person on the ground able to attend to other tasks. Conversely, if the person on the ground had to send the command to each spacecraft individually, wait for each to send an image down, and then filter the data, the image would be so late in coming that it may render the image unnecessary. Figure 1 shows a 3-dimensional visualization of the five spacecraft constellation used in the Sensor to Shooter example.

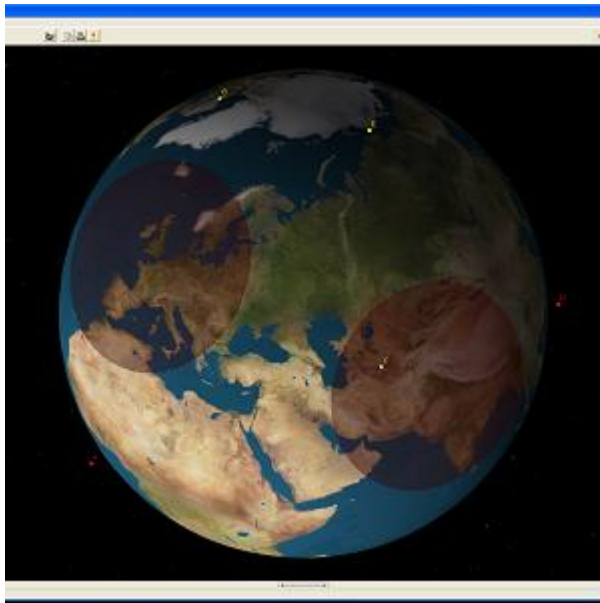


Figure 1: Sensor to Shooter using FreeFlyer®

### **CURRENT LIMITATIONS IN EXISTING RESPONSIVE SPACE SOLUTIONS**

In the discussion of the capabilities of current autonomy software tools, it is important to be aware of some of the key limitations. The first of these may be the classified nature of

many of the data products generated on-board. Many of the generated products can be more sensitive than their constituent raw data and must therefore only be handled on the ground with appropriately classified networks. This may unintentionally limit the distribution of the classified product(s) with an appropriate level of decryption capability (in addition to the normal RF receiving equipment).

Similar restrictions are present on the commanding capability of the users. Almost all military missions require encrypted uplinks, but the easy to use nature of autonomous spacecraft may require even more rigorous authentication of commanding authority, along with a standard method of establishing command priority when multiple command requests may be possible.

Using existing methods to communicate with the SIPRnet, Special Operations Forces (SOF's) may send a task to an autonomous asset either by using their SIPRnet access to send a message and rapidly relocate, or use tight-beam methods to uplink a command directly to the autonomous orbital asset. The latter scenario would require SOFs to know the location of the orbital asset or relay satellite and have the appropriate gear (more burdensome). In the former scenario, once the tasking request is sent via SIPRnet or equivalent, the in-theater Common Data Link (CDL) uplink station can send the command to the orbital asset as it comes in view. CDL stations are currently used to operate a variety of unmanned air vehicles.

Tasking requests received by the autonomous spacecraft are evaluated based on priority, orbital geometry, and spacecraft operating constraints. For the SOF use case, one danger is that in the course of performing their mission, each SOF unit in theater may view their situation as highest priority, leading to potential for multiple highest priority requests.

An alternate method is to have the theater commander, who is coordinating the various units that may request spacecraft tasking, assign non-sequential priority codes to each SOF unit for each particular mission. This way the SOF units would not be able to label multiple requests as high priority.

On-board software can also vary a more extensive list of parameters making it harder for requests to be labeled with the same priority. This would allow more decision making on-board and less end user interaction on the ground in making the priority decisions.

Once a data product is available, low bandwidth versions can be downlinked via omni-directional broadcast to prevent revealing the SOF unit location. Low bandwidth products could simply be small text files with target information and coordinates or small jpeg files of target image chips or thumbnails. Meanwhile the full resolution data products are sent down to the in theater CDL groundstation for retrieval via secure network connection by any authorized party.

Another limitation to command and telemetry issues with autonomous software is the data capacity of the particular terrestrial secure network that the commands and end products will be routinely sent through. The various networks have their own individual capacities and features and the ground system architect must be aware of limitations to data volumes, reliability of service, latency, maintenance requirements and geographic availability for the particular mission under consideration. It is also important to consider off-nominal network use that may be required during these exercises, such as the demands that may be placed on the system or the possibility of periodic calibration or map uploads.

### **THE IMPACT OF SPACECRAFT AUTONOMY ON INTEGRATION AND TEST**

Significant time and cost savings can be realized in the preparation for integration and test by wisely planning the capability of the autonomy system and structuring the integration and test plan to take full advantage of that capability. For instance, one of the principle functions of spacecraft autonomy for emerging missions is the ability to rapidly perform a self-test upon separation after launch, and then to perform as much of the on-orbit commissioning of the spacecraft as possible. This can include performing deployments, placing the vehicle in a power safe attitude, powering on and checking out each individual component of the spacecraft, and performing initial calibration and alignment activities for the attitude control system and the payloads.

In order to realize cost and time savings in integration and test (I&T), it is necessary to coordinate development of the autonomous checkout software and development of the verification process used on the ground. By aligning these two functions as closely as possible significant savings can be realized in developing the process of verifying spacecraft functionality on the ground and in space. In addition, by utilizing flight autonomy software, similar to an automated FreeFlyer® or AutoCon™, during ground checkout, operators and test engineers will get important practice in the “testing like you fly” principle. This gives operators great insight into how the vehicle will truly behave and gives the autonomy software developers terrific insight into what features and functions are especially useful to ground operators.

To maximize the overlap between objectives the development team should strive to construct a verification plan which is

amenable to verification via automation. This means designing functional elements of the spacecraft to test in an explicit manner without requiring extensive reconfiguration and without convoluted analysis. One simple way to support this is to ensure that the spacecraft telemetry includes the appropriate test points as explicitly telemetered values.

Since it is unlikely that 100% of all I&T testing will be replicated in the on-orbit autonomy, there will be some need for separate tests for both of these areas. To leverage a common tool set of modular tests to support these separate goals, it is important that the I&T team and the autonomy software team agree on a common software style and toolset early in the development cycle. This will allow developers to share modular elements using the same script engine for both ground and flight activities if possible. To facilitate this, careful consideration should be given to choose a scripting language which is amenable to inclusion in flight software. Examples of features to avoid would be languages which require a non-real time operating system or those that have an excessively large memory footprint.

In the case of NASA's EO-1 mission and AutoCon™, the full graphical user interface of FreeFlyer® was removed and only the computational back end engine was used on-board. Using this approach, a much smaller memory footprint is achieved. Because AutoCon™ is essentially the same tool as FreeFlyer®, without the interface, the ground version of FreeFlyer® can be used to verify the on-board tool. Testing is then simplified because the same algorithms are used in both tools.

For the I&T team to make use of autonomy during testing, it is necessary to develop the autonomy software early in program so that it is ready to use when hardware starts arriving.

The advantage of developing the software early, to the autonomy team, is that the product gets thousands of hours of test time on the spacecraft during development on the ground, and input from the test team to ensure it adequately performs requirements verification. As a result, it is comparatively easier to produce software which does a more thorough job of functional and performance checkout. In addition, with all of the practice running operational scenarios on the ground, the autonomous tasking features will become highly refined and streamlined by the time they reach orbit, leading to a high likelihood of success. By testing the software on the ground with the spacecraft bus prior to the mission, the safety of the spacecraft is also ensured. System tests with end users, throughout the design of the software, will help with overall design usability.

Early development of autonomy in the program life cycle also brings the end user inputs into the development early. This benefits the program by increasing the final product's relevance to the end user and ensures that key features are identified early in the program when they are relatively easy to add. Also, telemetry and command screens designed at this stage for I&T can incorporate features desired by the end user and therefore serve as the operations system in something close to final form. This makes the transition from test to operations easier and more straightforward for I&T personnel to serve as part of the operations team.

The I&T team is also forced to conduct tests in a more integrated fashion. This results in testing which more accurately simulates on-orbit conditions. This ensures that higher level integration issues are worked early leaving less room for surprises at the end of integration. By using the autonomy software early and coupling it with end to end tests including field communications gear or test

sets, the team can verify the full functional operations path at an earlier stage. These features combine to shorten the overall time required by the test team.

### **INDUSTRY-WIDE IMPACTS OF SPACECRAFT AUTONOMY**

The most interesting impact of autonomy from the responsive space community's viewpoint is the shorter development schedules that this technology enables. For mature spacecraft types, such as the OpSat's being envisioned as TacSat successors, mature autonomy software will be the key technology in enabling rapid fielding of new assets. The only way to realistically achieve one week capability from call-up to on-orbit operation is to rely very heavily on automated testing to verify that the spacecraft is ready to launch and to commission the spacecraft once it is on-orbit. Only autonomy software can perform the thousands of checks necessary in the requisite time. Because of the amount of customization necessary for research and development spacecraft, the development of associated autonomy checks requires a reasonable amount of time. Conversely, their early adoption usually results in significant time savings in the later stages of the program.

Creating detailed system requirements early in the design of the ground, on-board, and spacecraft systems from the end user will make the entire system more functional. Using the experience of ground operations, the spacecraft on-board software can be configured with as many changing variables as possible. Ground experience will also help to determine the ideal size of the constellation for responsive space. The end user can determine exactly what they would like to see from the ground perspective. The added benefit of a flexible COTS software solution, like FreeFlyer®, is that if parameters need to

be changed, added, deleted, or updated, they can be without help from the vendor.

Perhaps the most fundamental impact that autonomy can have on the responsive space market is the shift in focus from the strategic to the tactical realm. Autonomy brings the ability to operate a spacecraft directly to the end user, with its simple interface designed for the non-expert. In addition, the capability to respond instantly to changing mission priorities allows autonomous spacecraft to play a key role in real time support of combat operations. The autonomy features give spacecraft an unprecedented flexibility to meet these changing mission requirements. These capabilities, coupled with small satellites' ability to be deployed rapidly, will put very powerful tools in the hands of front line and unit level commanders, drastically changing the utility of space assets for the responsive space market.

### **BUILDING THE AUTONOMOUS SOLUTION FOR RESPONSIVE SPACE**

In order to build the most effective software solution for autonomous space, engineering experts, ground operators, and end user communities need to come together early in the design cycle of the system solution. To effectively develop the next generation autonomy software use case scenarios must be developed early on by expert and non-expert end users. Spacecraft bus designers along with software developers and experts in operations must come together to design, build, and test a unified product for a responsive space solution.

### **CONCLUSIONS**

Autonomy software not only reduces the amount of time and effort necessary to develop and maintain spacecraft, but it changes the fundamental market that these

products can serve. By enabling incredibly short call up times, direct tasking from the end user, and data product delivery directly to the end user, small satellites can play a greatly expanded role on the tactical battlefield. By putting these very powerful tools including data analysis and processing on-board the spacecraft, we dramatically enhance the situational awareness of front line combatants.

a.i. solutions, has over ten years of expertise in spacecraft mission services and software. MSI has experience with the design and testing of the spacecraft bus. Using this expertise and knowledge of the end user, we can make an autonomous system that is safe and easy to use.

Using the COTS product FreeFlyer® along with a.i. solutions, Inc. expertise in mission design and operations, MSI can incorporate on-board software to their spacecraft design and test. The cost savings of using a software solution that is already a proven product is drastic over the cost of designing an entirely new product or using legacy products that are expensive to keep updated.

### **ACKNOWLEDGEMENTS**

Special thanks to Daryl Carrington of a.i. solutions, Inc.