Demonstrations of System-Level Autonomy for Spacecraft

**Martin S. Feather, Brian Kennedy, Ryan Mackey,
Martina Troesch, Cornelia Altenbuchner,
Robert Bocchino, Lorraine Fesq, Randall Hughes,
Faiz Mirza, Allen Nikora, Patricia Beauchamp
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
{Martin.S.Feather, Brian.M.Kennedy,
Martina.I.Troesch, Cornelia.Altenbuchner,
Robert.L.Bocchino, Lorraine.M.Fesq,
Scott.Hughes, Faiz.Mirza,
Patricia.M.Beauchamp}@jpl.nasa.gov
anikora@alumni.caltech.edu**

**Patrick Doran
Pearl River Technologies, LLC
NASA Goddard Space Flight Center
8800 Greenbelt Rd.
Greenbelt, MD 20771
pldoran@cpp.edu**

**Ksenia O. Kolcio, Matthew J. Litke, Maurice Prather
Okean Solutions, Inc.
1211 E. Denny Way, #32A
Seattle, WA 98112
{ksenia, matthew, maurice}@okean.solutions**

Abstract—**System-level autonomy refers to autonomously meeting the crosscutting needs of a system through awareness and coordinated control spanning the system's breadth of capabilities. In contrast to function-level autonomy, which focuses on capabilities required to achieve a specific function such as surface navigation or image recognition, system-level autonomy addresses the needs to coordinate and manage activities and resources, and estimate the state, across subsystems.**

This paper describes demonstrations that were conducted on a spacecraft workstation testbed. The autonomy was provided by system-level planning and execution integrated with system-level estimators of orbit knowledge and spacecraft hardware health. These components are embedded in a system-level framework defining how goals are formed and executed, which elements exist, and how control authority is distributed among components. The planning and execution system at the heart of the framework has the capability to schedule, execute and monitor completion of tasks, as well as plan around unexpected events including new science opportunities and anomalies. The planning and scheduling system is the Multi-mission EXECutive (MEXEC), supported by the system-level health state estimator Model-Based Off-Nominal State Identification and Detection (MONSID), and Autonomous Navigation (AutoNav) algorithms, which determine the orbital system state based on optical observation of other targets. These components are applicable to many kinds of missions on different platforms.

These demonstrations were elaborations of earlier experiments conducted on the ASTERIA (Arcsecond Space Telescope Enabling Research In Astrophysics) CubeSat, described in a companion submission [1]. The spacecraft's extended mission served as an in-flight test platform, during which some individual autonomous capabilities were flown successfully. The autonomy experiments described here were performed on the ASTERIA workstation testbed.

TABLE OF CONTENTS

1. INTRODUCTION

Achieving the bold missions that lie in NASA's future will hinge on the use of increasingly autonomous systems. This direction has been identified in past studies, including the 2018 Workshop on Autonomy for Future NASA Science Missions [2], and just recently in [3]. These studies foresee the need for on-board autonomy to take over more control of a mission's activities. But, if this is to come to pass, people must have confidence that the autonomy can be developed, deployed, and thereafter perform as intended when in control of NASA's valuable assets.

This paper reports on autonomy demonstrations that take a step towards instilling such confidence. The demonstrations are of a combination of several autonomous capabilities, each designed to be multi-purpose, meaning they are not crafted for only a specific application but can be readily customized for a wide variety of missions and activities within those missions. Furthermore, the autonomous capabilities have been developed following JPL's flight programming practices, for example, their software has been written to meet JPL's software coding standards.

The significance of these demonstrations is twofold: they increase confidence in the correct operation of the three individual components, MEXEC, MONSID and AutoNav, and they show the capabilities of them acting in combination. The demonstration scenarios transformed orbit determination from a housekeeping task that falls upon ground operations to repetitively perform, to an on-board autonomous solution that, once started, maintains itself

2021 IEEE Aerospace Conference | 978-1-7281-7436-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/AERO50100.2021.9438427

978-1-7281-7436-5/21/$31.00 ©2021 IEEE

henceforth. Furthermore, this solution was shown to be robust in the face of faults, able to distinguish when a fault did not impact current activities and thus allowed them to continue unperturbed, as well as when a fault undermined the integrity of information on which some activities depended, leading to canceling of those activities. The demonstration scenarios also showed how on-board activities are rescheduled once anomalies have been removed, and conditions once again are safe to proceed.

The demonstrations illustrates several scenarios of autonomous determination of orbit knowledge, needed to maintain its accuracy sufficient for science purposes. Orbit knowledge is determined by the navigation system identifying images to take of relatively nearby objects, from which the spacecraft's orbit is more precisely recalculated. Because orbit knowledge accuracy decays over time (due to influences of magnetic fields and tenuous atmospheric drag), it is necessary to perform this determination prior to its decay below a level needed for science observations.

Several benefits of on-board system-level autonomy are illustrated by these demonstrations:

- Relieving the burden of repeated cycles of ground control by having the on-board planning and execution system autonomously maintain sufficient orbit knowledge accuracy.
- Avoiding conflict with all higher-priority activities such as communication passes when scheduling the orbit determination activities, made possible by the planning and scheduling system-wide awareness of all plans and conditions.
- Ability to "fail operational" by recognizing and responding appropriately to off-nominal events. This avoids the need to resort to "safe mode" and requiring operators' help in all but the most unlikely off-nominal scenarios. Included in the demonstrations are both nominal scenarios, during which all the components function as intended, and off-nominal scenarios during which disruptions caused by transient or repairable anomalies are recognized and reacted to, autonomously.

The remainder of this paper is organized as follows:

- A discussion of the distinction between system-level and function-level autonomy, with examples of each, is given in the section 2, "Related Work".

- Section 3, Background, briefly introduces the demonstration milieu – the testbed of an Earth-orbiting spacecraft, available for demonstrations purposes following the end of mission. Each of the three autonomy capabilities are then summarized.

- Section 4 describes sufficient details of MEXEC, to understand how it is used in the demonstrations to control spacecraft activity.

- Section 5 presents the demonstration scenarios, an Earth-orbiting spacecraft autonomously calculating its own orbit.

- Section 6 gives the results of the demonstrations, in which autonomous orbit determination was done first in a fault-free test, and then in tests in which faults were deliberately injected to see if the autonomy would react as intended.

- Section 7 offers conclusions.

2. Related Work

System-level autonomy refers to autonomously meeting the crosscutting needs of a system through awareness and coordinated control spanning the system's breadth of capabilities. The Remote Agent Experiment (RAX) [4], which flew on NASA's Deep Space 1 (DS1) spacecraft, was the first in-flight demonstration of autonomy addressing system-level needs. [5] describes the experiments done in May 1999 during which the Remote Agent (RA), an Artificial Intelligence based closed loop autonomous control system, was allowed to take control of the DS1 spacecraft.

Included in the RA architecture were:

- a model-based planner/scheduler, the Heuristic Scheduling Testbed System [6]; its use in RAX is described in [7].

- a model-based Mode Identification and Recovery (MIR) engine "Livingstone" [8].

- a robust multi-threaded executive [9]. This executive carried out plans provided by the planner, and responded to fault diagnoses and repair suggestions from MIR. As it did so, it ensured that the activities in the plan would be dispatched and terminated at times consistent with the temporal constraints in the plan.

The conduct of the experiment is described in [10]. Two scenarios were executed in which RA was allowed to control the spacecraft. The experiment demonstrated nominal operations with goal-oriented commanding and closed-loop plan execution, and fault management capabilities with failure diagnosis and recovery, on-board.

The demonstrations reported in this paper are based on similar capabilities. However, our approach is distinguished from RA by a stronger focus on compatibility with other spacecraft software and systems, enabled by the FRESCO reusable spacecraft autonomy architecture described in a companion submission [11]. This structured approach to architecting an autonomous system provides a path to infusion on future missions, whereas the Remote Agent, while successful in its in-flight demonstrations, did not lead to such uptake.

Other notable work on system-level autonomy includes the Autonomous Sciencecraft Experiment on the Earth Observing One (EO-1) spacecraft, which proved successful in its ability to detect dynamic science events, including volcanic eruptions, flooding, ice breakup, and cloud cover change [12]. This was through use of the onboard Continuous Activity Scheduling Planning Execution and Replanning (CASPER) software [13], whose plans were passed as inputs to the Spacecraft Command Language (SCL) executive system to generates the corresponding detailed command sequences. For leads to more work of this nature, see [14], which surveys model-based techniques as the means to provide autonomous on-board capabilities for spacecraft, and describes operational concepts for mission planning and execution in European space projects. [15] is a more recent and extensive survey of spacecraft on-board planning and scheduling, listing many examples and comparisons among the approaches they exemplify.

In the area of robustness and fault protection, in practice most spacecraft rely on detecting hardware component failures and switching to redundant elements if available, or going into safe mode and contacting mission control for help. An example is the New Horizons Pluto Mission, with its high reliability requirements and long light-time from Earth. This spacecraft used onboard software in the form of a rule based system to detect faults and then trigger macros to perform the appropriate reconfiguration of its highly redundant hardware [16]. Another example is the BepiColombo mission to Mercury. As described in [17], the FDIR system's roles are to maximize system availability for normal operations and to safeguard the spacecraft at all times, notably to control adherence to strict attitude constraints, including in safe mode. Because of the length of time a reboot of the onboard computer would take and the urgency of correct attitude control, FDIR is backed up by "Failure Control Electronics" to perform the critical attitude control function.

Work towards more capable fault detection, using a combination of both model based reasoning and machine learning, is reported in [18]. The focus of this work is on the fault detection and identification (of the faulty component), information to then be used to direct the appropriate response, which the authors acknowledge in the case of decreased capabilities, will require replanning and rescheduling.

A different approach, one that avoids the use of onboard planning and scheduling and yet is responsive to (some) unpredictable variations in conditions is described in [Gillette et al], through using "*a combination of constraint and priority parameters associated with every task to ensure robust task execution with behavior as intended*." The authors acknowledge that there can be unexpected situations for which their lack of an onboard planner would mean the spacecraft would have to default to safe mode.

The desire to use automation both to make spacecraft less demanding of operators' time and attention, and to make them robust to faults and failures, is widespread. For example, [19] describes research aiming to achieve ESA's E4 level of mission execution autonomy [20], in which on-board goal-oriented mission operations performs re-planning when anomalies arise.

Function-level autonomy

There have been numerous applications of "function-level autonomy" in many NASA missions. Some examples are listed next:

- The Descent Image Motion Estimation System (DIMES) determined horizontal velocities imparted by cross-winds in the Martian atmosphere to direct thruster firings to compensate if need be. To make this determination it performed image correlation between locations within successive images taken of the surface as it rapidly descended. [21]. More recently, Terrain Relative Navigation (TRN) has made its way into the Mars 2020 program, where it will be used to guide the landing of NASA's Perseverance rover – see https://mars.nasa.gov/mars2020/timeline/landing/entry-descent-landing/

  DIMES itself estimated the plausibility of its results by cross-checking the acceleration derived from its velocity estimates with that measured by the spacecraft's IMU. This illustrates a special-purpose implementation of health estimation built as part of the function-level autonomy.

- NASA's Curiosity rover, in the first seven years on the surface of Mars, drove over 21 kilometers. As described in [22], "*Curiosity has driven over a variety of terrain types and slopes, employing multiple drive modes with varying amounts of onboard autonomy including Visual Odometry, Hazard Detection, Hazard Avoidance, and Visual Target Tracking*".

  These autonomous capabilities combine to allow the rover to plan safe routes, and to always drive safely when following paths whether planned by mission-control or itself. For example, visual odometry allows the rover to measure changes in its position during a drive, a use of which is to have the rover halt if pre-established slip thresholds are exceeded. These functional capabilities have been engineered and verified for the specific challenges of surface traversing, and proven effective in the mission to date. However, their general reaction to an anomaly is to halt the drive (or to not even start it if initial conditions are not suitable) and wait for ground control to assess the situation and direct the rover accordingly. Other than low-level exceptions (e.g., scrubbing memory errors), there is no system-level autonomy attempting to recover and continue.

- The Autonomous Exploration for Gathering Increased Science (AEGIS) system, as its name suggests, increases the return of science data. It was fielded first on NASA's Mars Exploration Rover Opportunity, and is now in operation on NASA's Curiosity Mars Rover. AEGIS is autonomous software running on the rover to identify scientifically interesting targets, pick the one with the highest science priority, and conduct a more detailed observation of that target [23].

  Although AEGIS itself exemplifies an advantageous use of autonomy, it functions as a well circumscribed capability. It is not self-scheduled, rather, ground control plans when it will be admissible to allow AEGIS to run at the conclusion of drive. It also makes no attempt to react to anomalies other than to "exit gracefully" by safely performing whatever cleanup is necessary if it halts or is interrupted by mission-level software.

- Autonomous navigation (AutoNav) in space was demonstrated first on Deep Space 1, and then used on several subsequent missions: on Deep Impact (DI), to provide terminal guidance to the impactor spacecraft to hit comet Tempel 1; on the same mission's parent spacecraft to track the nucleus of Temple 1 as it few by, and on Stardust's comet flyby [24]. AutoNav is part of the demonstration reported in this paper, but in this case it performed orbit determination while circling the Earth.

  The successes of AutoNav have led to its use during critical in-space maneuvers when distances are such that light-time delays preclude mission control supplied directions. This demonstration has a different emphasis, described next, and as such relies on use of system-wide autonomy capabilities.

3. BACKGROUND

ASTERIA (Arcsecond Space Telescope Enabling Research in Astrophysics) was a technology demonstration mission to conduct astrophysical measurements using a CubeSat. Its successful primary mission [25] was followed by three extended missions, during the last of which it served as a platform to demonstrate several autonomous capabilities as well as continuing science observations [26] as an opportunistic exo-planet science platform. Once the mission ended, demonstrations continued on ASTERIA's testbed, culminating in the demonstrations described herein.

JPL's F' smallsat software [27] was the flight software on ASTERIA, and each of the autonomy capabilities were hosted inside this software framework. These capabilities are described next.

AutoNav

Autonomous navigation ("AutoNav") was one of the several new technologies successfully demonstrated on the DS1 mission [28]. It determined the spacecraft's location during cruise based on images of asteroids and stars taken by the spacecraft's integrated camera and spectrometer. Since then it has been used for critical course control on several flybys of comets and the impact of Deep Impact's impactor spacecraft [29].

A demonstration in 2019 of AutoNav on ASTERIA showed it could be used on a CubeSat to determine Low Earth Orbit (LEO) trajectory details from passive imaging alone, using images of an asteroid (Vesta). To perform this experiment, ASTERIA commanded AutoNav with a traditional command sequence uploaded from ground control. The intent had been to perform an on-board demonstration of orbit determination from Geosat imagery early in 2020, but in December 2019 contact with ASTERIA was lost, and never recovered before the spacecraft de-orbited [30].

MONSID

MONSID (Model-Based Off-Nominal State Identification and Detection) is software developed by Okean Solutions Inc. that determines the health status of a system's components by propagating sensor data through a nominal model of the system's behavior [31]. An inconsistency between the model and the sensor data indicates an off-nominal condition. MONSID implements a technique called "constraint suspension" [32] to identify the component or set of components whose behavior constraints are inconsistent with the hardware's actual behavior, and thus account for the system's off-nominal behavior.

An advantage of MONSID's approach is that it requires only a model of nominal operation. This is in contrast to some other methods for health determination, which depend on the need to model the observable symptoms of off-nominal behavior should faults occur. As a consequence, MONSID is able to detect faults that system engineers have not predicted – this was seen in recent experiments in which MONSID was deployed on a test rover being put through its paces at JPL [33]. A further advantage is that testing MONSID's model can make use of the same tests that would be designed for the system itself [34].

The MONSID engine, written to meet JPL's software coding standards, and a constraint-suspension based model of ASTERIA's attitude control system successfully underwent the flight readiness reviews necessary for in-flight testing on the spacecraft. However, a month prior to uploading the software, ASTERIA stopped communicating. This loss of communication meant that MONSID's in-flight experiment was no longer possible, and testing instead continued on the ASTERIA testbed using high-rate attitude control system data already captured from ASTERIA, described in a companion submission [35].

MEXEC

MEXEC (Multi-mission EXECutive) is autonomous planning and control software, described in [36] as "a lightweight on-board planning and execution system that monitors spacecraft state to robustly respond to current conditions." MEXEC was designed for a wide range of mission uses (e.g., on spacecraft, rovers, landers) and to be compatible with a variety of flight software and operating systems.

Prior Experiments with MEXEC—During the Europa mission's Jovian orbits the spacecraft will experience the high radiation at Jupiter, and hence the probability of flight software resets is high. With this in mind, an MEXEC prototype was developed to have the capability to restart the science plan after flight software resets [36]. As described there, "The development was done as part of Europa-focused flight software, but MEXEC was designed to be applicable to landers and rovers as well." The prototype was demonstrated in Europa flyby scenarios in the Europa flight software test environment.

MEXEC was developed further in the 2017 – 2019 timeframe, and demonstrated in the DARTS software simulation [37] augmented with GN&C capabilities and models of a spacecraft and its components. This was done as part of an effort taking a broader perspective on architecting autonomous systems, another outcome of which was the Framework for the Robust Execution of Spacecraft Commands On-board (FRESCO) [11]. This framework describes the general components and interfaces of autonomous systems, and how compliant architectures can be designed. MEXEC is an instance of a compliant architecture.

In the summer of 2019, the first experiment to have MEXEC execute on-board and control a spacecraft, ASTERIA, was successful [26]. For this, the MEXEC software was integrated with the existing ASTERIA flight software, and then used to execute tasknets on-board to perform nominal operations of setting up passes and taking science observations. This showed that MEXEC could replicate the behavior of standard sequences.

A second in-flight experiment was planned, but communication with the spacecraft was lost before it could be performed on-board, so it was conducted instead on the ASTERIA testbed. In this experiment, MEXEC was used to manage momentum of the reaction wheels of the spacecraft's attitude control system. Traditional commanding had addressed this while ASTERIA was still in flight, by ground control predicting momentum buildup in advance, leading to the scheduling of frequent momentum-dumping activities [38]. The experiment showed that MEXEC could take over this role, leading to momentum-dumping only when necessary, thus leaving more time for science observation [26].

MEXEC supports autonomy by using task or goal-based planning and commanding, maintaining intentions and effects on board the spacecraft. This is in contrast to the prevalent form of spacecraft commanding through sequences, which lack information about their purpose. The next section provides more details of MEXEC necessary to follow how it accomplished control of the demonstrations.

4. DETAILS OF MEXEC

The MEXEC software is described in detail in [39], from which the following summary and figure are taken:

Figure 1 shows a diagram of the MEXEC modules and their interactions with the ground and other flight components. MEXEC assumes that there is a state database that reports system state and a command dispatcher that dispatches commands. The ground sends a tasknet to the spacecraft, which is read by the planner. The planner schedules the tasks with the help of the timeline library and the system state from the state database. Once sufficiently close to the start time of a task, the planner passes it on to the executive, which performs real-time constraint checking to ensure safe execution of the task. Task execution updates are sent from the executive to the planner to keep the planner informed in case re-planning is necessary.

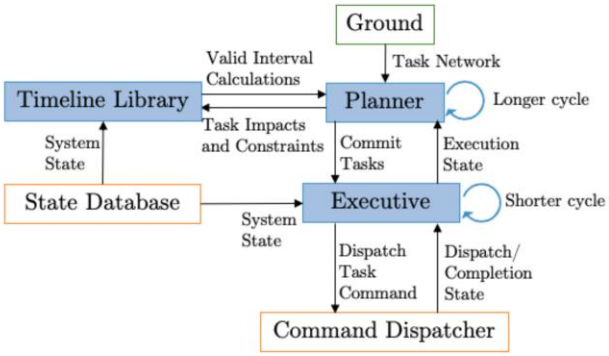


Figure 1 – MEXEC's components (in blue) and the interactions with and between them [39]

The brief descriptions of MEXEC's key concepts that follow are taken from that same source, which also provides a discussion of related work on more flexible commanding (e.g., using the Virtual Machine Language (VML)) and on other instances of spacecraft "executives" to run these more flexible forms of commands.

Tasks—An MEXEC Task expresses spacecraft behavior together with constraints on when the task may be executed. A task may provide a spacecraft command or command sequence to be dispatched, i.e., sent to the spacecraft to perform some activity, when the task is executed. There are two kinds of constraints that may be associated with a task: a pre constraint, which must hold for the task's execution to begin, and a maintenance constraint, which must remain true throughout the task's execution (if it becomes false

before the task completes, the task's execution halts, and the task is said to have "failed"). Also associated with a task may be impacts, expressing the changes expected from execution of the task. For example, the impact of execution of a task that dispatches a command to turn on a spacecraft's heater would, provided the spacecraft is operating correctly, be expected to turn the heater on and increase the electrical power being drawn. Impacts are the means by which the planner can know the effects that tasks are expected to have, and thereby schedule them accordingly. In addition, the successful execution of a task may be used to set values on "internal states" within MEXEC, as may its failed execution, including when a cleanup command is used. These internal states, and the tasks' effects on them, are visible to the planner for its purposes of scheduling, and may be referenced in tasks' pre constraints and maintenance constraints. Internal states are used when it is not possible to include the state in the state database (because of time limitations for a demonstration, or because it isn't a state monitored by the system), and/or when we want everything to behave as if everything executed as expected.

The above is not a complete list of tasks' features available to MEXEC; others not used in this demonstration, such as the ability to retry a failed task's commands, exist but are not elaborated in this paper.

Timelines—Each spacecraft state or resource that is referred to in any task's constraint or impact is represented in its own timeline. Values reported by the spacecraft system to the state database are applied to the appropriate timelines to keep their values current. The planner uses task impacts to project timelines into the future so as to schedules tasks to satisfy all their constraints, as well as global constraints.

Tasknets—MEXEC tasks are typically organized into structures called tasknets (also referred to as task networks). These could be as simple as a linear sequence of tasks, or, as will be seen later, organized hierarchically. An MEXEC-controlled spacecraft can be commanded from the ground by uploading tasknets for MEXEC's Planner to schedule. Tasks can also be created on-board by MEXEC's planner by instantiating a *template*. Once a task has been scheduled, it is executed by the MEXEC Executive when the time to do so has arrived.

Templates—templates for tasks are available to the planner to, if necessary, instantiate with details and schedule.

5. THE DEMONSTRATIONS

The goal of the demonstrations described in this paper was to show how MEXEC uses the health status information provided by MONSID to manage the AutoNav software and its related activities though both nominal and off-nominal (when faults occur) conditions.

Scenario

The demonstration's mission scenarios setting is as follows. ASTERIA's science observations require ASTERIA's orbit to be known to within a predetermined accuracy. Over time, uncertainty in this knowledge grows due to unpredictable magnetic and gravitational influences affecting the spacecraft's trajectory. AutoNav is used to improve orbit knowledge by taking images of other orbiting spacecraft whose positions are known, and from their locations in the images, AutoNav calculates ASTERIA's orbit to a high degree of accuracy (Figure 2). AutoNav needs only approximate knowledge of ASTERIA's orbit to direct the pointing of the camera in the general direction of known orbiting geostationary spacecraft, ensuring they will be in the field of view. AutoNav's accurate determination of orbit from images taken of them is achievable because those other orbiting spacecraft, in contrast to more distant targets such as asteroids or star constellations, offer high parallax from which to make the determination.

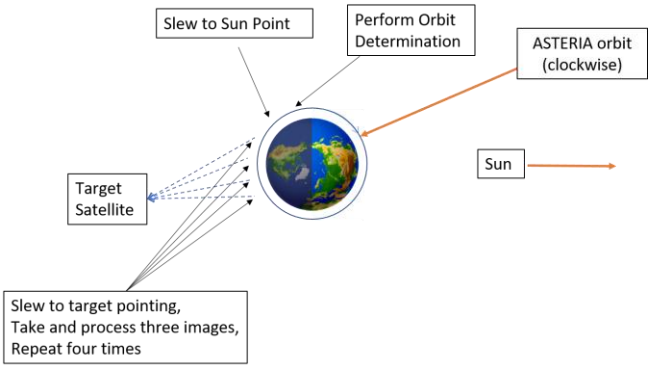


Figure 2 – AutoNav gathers images for orbit determination while the Sun is eclipsed by Earth

AutoNav's imaging and recalculation of orbit takes place over three orbits during times in which the Sun is eclipsed by Earth. During each eclipse a) the spacecraft's attitude is set to point the camera in the direction of another orbiting spacecraft, b) three images are taken of that spacecraft over the time it is predicted to remain in the camera's field of view, and c) pointing and imaging is repeated three more times. At least two eclipses' worth of imaging are necessary for AutoNav to sufficiently improve the accuracy of orbit knowledge. If all activities perform perfectly, analyzing the images obtained during a third eclipse would make little further improvement to orbit knowledge, but planning for a third eclipse adds robustness to vagaries of imaging and image processing, and, as we will see, robustness to faults that can disrupt AutoNav's activities. Even more robustness could be had by planning to continue in a fourth eclipse, but this would in most cases be a waste of resources, and engineering judgement led to the decision to plan orbit knowledge improvement over three eclipses.

The steps of this scenario are portrayed graphically in Figure 3, the elements of which are as follows:

"*Improve Orbit Knowledge*" is the overall activity by which MEXEC's knowledge of the spacecraft's orbit is improved. It takes the form of a hierarchical task, and is done over three "*Image Sessions*" each of which takes place during a separate eclipse; the last one is followed by "*Push OD (Orbit Determination) info.*," which transfers AutoNav's determination of the spacecraft's orbit to MEXEC. Each *Image Session* comprises the following steps:

- "*Camera ON*," as its name suggests, turns the camera on ready to take images
- "*Go to Att.*" is the action to slew the spacecraft to point in the direction of the target, another orbiting spacecraft
- "*Take & Proc. x 3*" is the repetition three times of "*Take & Proc.*," which comprises the following steps:
  - "*Take Image*" is the camera exposure activity over a pre-determined length of time
  - "*Buffer->AutoNav*" transfers control of the image buffer to the AutoNav software, and the later "*Buffer->Camera*" returns control.
  - "*Transform Image*" and "*Process Image*" are AutoNav's processing of the image data to contribute towards its later "*Orbit Determination (OD)*".
- "*Camera OFF*," as its name suggests, turns the camera off
- "*Go to sun.*" is the slew to sunpoint, to be in the attitude that points the solar panels in the direction of the sun when ASTERIA emerges from eclipse.
- "*Orbit Determination (OD)*" is AutoNav's determination of the spacecraft's orbit based on its orbit estimate prior to the *Image Session* augmented by the processing conducted on images taken during the current *Image Session*.

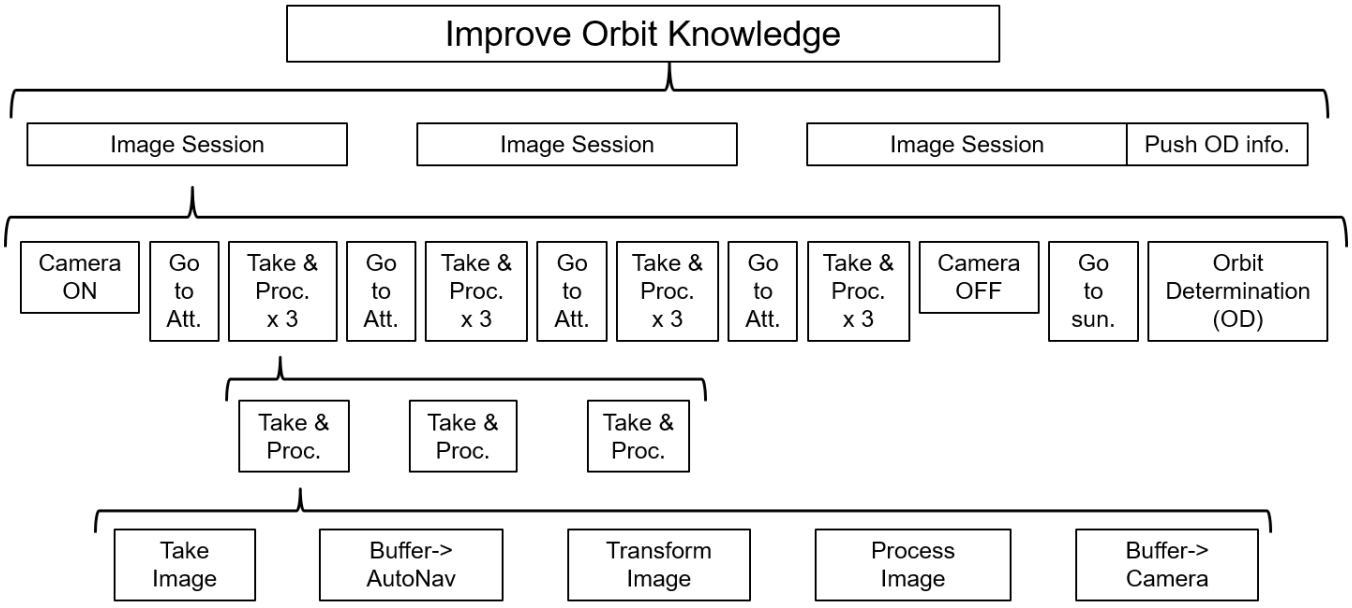


Figure 3 – *Improve Orbit Knowledge* tasknet structure

Scheduling the Improve Orbit Knowledge activities

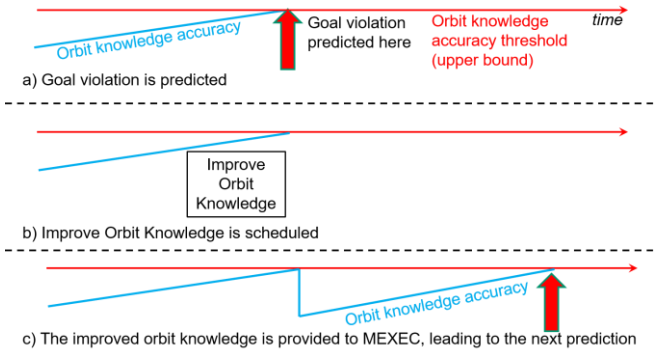


Figure 4 - Scheduling of *Improve Orbit Knowledge*

Orbit knowledge accuracy decays over time (due to influences of magnetic fields and tenuous atmospheric drag). Hence it is important to schedule *Improve Orbit Knowledge* to complete prior to knowledge decay, to stay within the threshold needed for science observations. In this demonstration, such scheduling is done automatically by MEXEC.

To make this possible, MEXEC's planner is provided a maintenance goal to keep the accuracy better than a predetermined threshold. The planner predicts orbit knowledge accuracy using the last information provided by AutoNav of orbit knowledge, an estimate of the rate at which certainty in that knowledge decays, and the times of upcoming eclipses. The planner projects this information to predict when the accuracy will worsen to above the

threshold. It is aware that the *Improve Orbit Knowledge* template has an impact to improve orbit knowledge, and so choses to turn that template into a task scheduled to complete prior to the time of the predicted goal violation. This is shown in Figure 4: prediction of the goal violation (a) leads to scheduling of *Improve Orbit Knowledge* (b), the result of which is an improvement in orbit knowledge accuracy, from which the prediction of the next goal violation is made (c).

When MEXEC's planner determines the need to schedule *Improve Orbit Knowledge*, it interacts with AutoNav to configure the *Image Session* tasks under *Improve Orbit Knowledge*, yielding a set of task instances for the MEXEC controller to execute when the time comes to do so. The steps in this interaction are as follows:

1. MEXEC's planner determines the three eclipses in which *Image Sessions* are to take place. It picks eclipses as late as possible before the accuracy threshold is breached, but avoiding conflict with any higher priority task already scheduled. For example, an eclipse in which the spacecraft is scheduled for a high priority communication with Earth would need to point the spacecraft for that purpose, and hence be in conflict with AutoNav's need to point the spacecraft towards geosats. Picking (non-conflicting) eclipses as late as possible leads to scheduling *Improve Orbit Knowledge* no more often than necessary, leaving as many other eclipses as possible available for the mission's science observations. MEXEC then sends AutoNav its selection of eclipses. The planner also overrides tasks with lower priority than *Improve Orbit Knowledge* when scheduling the latter's activities .
2. Based on MEXEC's selected eclipses, AutoNav computes the imaging to be done in each of them, and sends the details back to MEXEC. These details are the times at which to point the camera, the directions (expressed as quaternions) in which to point the camera, and the times and exposure durations to take images.
3. With this information MEXEC instantiates the subtask templates to complete the decomposition of the *Improve Orbit Knowledge* activity.

Scenario Dependencies

The *Improve Orbit Knowledge* task depends on the following components and resources to be available and operational at various times during *Improve Orbit Knowledge*:

- ASTERIA's camera to take the images, and the camera buffer to transfer images to AutoNav,
- ASTERIA's attitude control system, XACT, to slew the spacecraft to point its camera towards the intended "target" spacecraft and hold that pointing steady as the images are taken,
- AutoNav software to process images to update its estimate of the spacecraft's orbit, and
- ASTERIA's CPU, data storage, and electrical power needed by the above.

The attitude control system must hold ASTERIA's pointing steady during imaging for AutoNav to be able to make use of the resulting image. A fault of the attitude control system during this time would likely disrupt the pointing, and therefore result in an image unsuitable for AutoNav's processing.

MEXEC controls the orbit determination activities through the *Improve Orbit Knowledge* tasknet. The key to the demonstration is that this tasknet contains information of all the conditions necessary for successful execution of orbit determination. In detail:

- The *Take Image* task has the pre constraint that the spacecraft's attitude be the direction specified by AutoNav towards the "target" spacecraft. It also has the maintenance constraint that the spacecraft's attitude remain in that direction. Together these constraints ensure that the task succeeds if and only if the image was taken while the camera remained steadily pointing in the intended direction. On ending, this task writes a value to an MEXEC "internal state" indicating whether it succeeded or failed.
- The *Buffer->AutoNav* task has the pre constraint that the preceding *Take Image* task executed and completed successfully, which is indicated by the *Take Image* task having set the success value in the aforementioned MEXEC "internal state". It too writes a value to another MEXEC "internal state" indicating whether it succeeded or failed.
- The *Transform Image* task has the pre constraint that the preceding *Buffer->AutoNav* task executed and completed successfully, which is indicated by the *Buffer->AutoNav* task having set the success value in the aforementioned MEXEC "internal state". It too writes a value to another MEXEC "internal state" indicating whether it succeeded or failed.
- The *Process Image* task has the pre constraint that the preceding *Transform Image* task executed and completed successfully, which is indicated by the *Transform Image* task having set the success value in the aforementioned MEXEC "internal state". It too writes a value to another MEXEC "internal state" indicating whether it succeeded or failed.
- The *Buffer->Camera* task has the pre constraint that the preceding *Process Image* task executed and completed successfully, which is indicated by the *Process Image* task having set the success value in the aforementioned MEXEC "internal state".

The success/failure values set in internal states by the above tasks are used as pre constraints to the following tasks, thus ensuring only properly processed images are provided to AutoNav's orbit calculations, illustrated in Figure 5. This is important because images taken in an incorrect direction, not held steady even if in the correct direction, or incorrectly transformed or processed, should not be used for orbit determination since doing so could lead to an incorrect result.

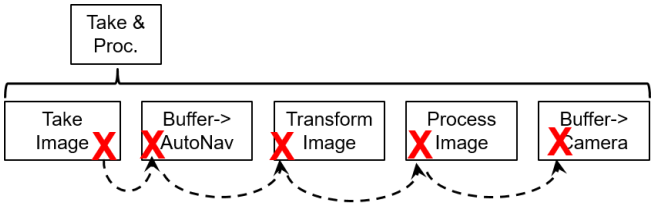


Figure 5 - Failure of *Take Image* leads to failure of the four tasks that follow; an image will not be processed

Detecting failure

As described above, the *Take Image* task has a pre constraint and a maintenance constraint both dependent on spacecraft attitude. Failure of the attitude control system during spacecraft slewing, or during holding the attitude steady during imaging, could violate these constraints. However, the attitude control system may have failed in such a way that it reports an attitude that appears correct while in practice it is incorrect. This is where the health status information that MONSID provides is useful. MONSID is used to continually monitor the internal behavior of the attitude control system, and detect whether that system is "healthy," i.e., the attitude it reports is accurate.

The *Take Image* task's pre constraint and maintenance constraint are refined to use the information MONSID provides:

- The *Take Image* task's pre constraint is that the spacecraft's attitude reported by the attitude control system be the direction specified by AutoNav towards the "target" spacecraft, and MONSID reports the attitude control system to be "healthy".
- The *Take Image* task's maintenance constraint is that the spacecraft's attitude reported by the attitude control system remain in that direction and MONSID reports the attitude control system to be "healthy".

The consequence of either of these constraints not being satisfied is that the *Take Image* task fails and the following *Buffer->AutoNav*, *Transform Image* and *Process Image* tasks also will fail, and thus there will be one less processed image for AutoNav to use to make its orbit determination.

When a failed *Take Image* task is within the last *Take & Proc.* before another *Go to Att.* (see Figure 6), that next *Go to Att.* will attempt to set the spacecraft's attitude, allowing future tasks that follow to have a chance of success.

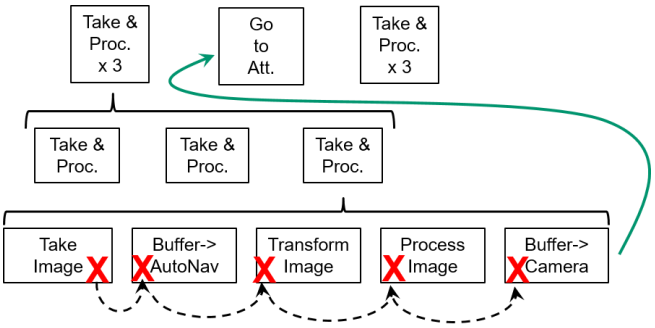


Figure 6 – Following the *Take & Proc.*'s failed tasks, a *Go to Att.* allows activities of the next *Take & Proc. x 3* to continue if the anomaly has cleared

When there is another *Take & Proc.* without any intervening *Go to Att.* (see Figure 7), then the *next Take & Proc.*'s *Take Image* pre constraint, that the spacecraft be pointed at the correct attitude, will depend on whether the attitude control system has recovered and pointed the spacecraft in the correct direction. If it is pointed correctly, and MONSID is reporting the attitude control system to be "healthy," than that *Take Image* will begin. Otherwise, the *Take Image* task will fail, the image will not be taken, and the image processing steps will be cancelled.

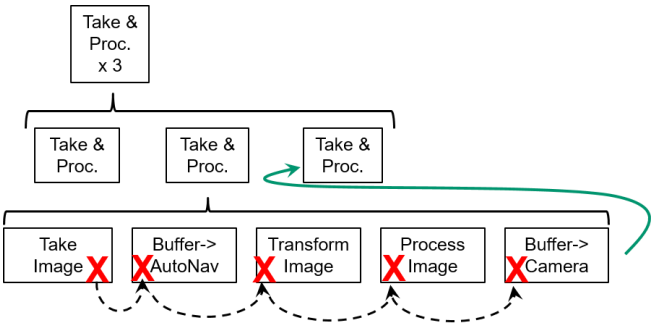


Figure 7 – another *Take & Proc.* after a failed *Take Image*

If after a transient failure the attitude control system does not automatically attempt to point back to the target to which it was previously pointing, it would require an explicit command to do so. MEXEC has the capability to execute a tasknet with such a behavior, but we chose to leave this for future investigation.

6. Demonstration Results

Our demonstrations were performed in two phases. First, MEXEC and AutoNav were integrated and run to execute the nominal (fault-free) Orbit Determination scenario without MONSID. Then MONSID was included, and the scenario was repeated with several faults deliberately injected at various times during the demonstrations.

Nominal Scenario (without MONSID)

The nominal scenario integrated MEXEC and AutoNav with the ASTERIA Flight Software, which is based on JPL's F', and executed on a desktop PC-Linux box. The timeline through which the health status of the attitude control system is communicated to MEXEC was present, but since MONSID was not included, its value remained set to indicate "healthy" status throughout the scenario.

The scenario was primed with the *Improve Orbit Knowledge* tasknet template, an initial estimate of eclipses, orbit accuracy and the rate at which accuracy decays. It was given the goal to maintain orbit accuracy below a threshold. MEXEC's planner predicted the future violation of this goal and identified the *Improve Orbit Knowledge* template as a means to preempt goal violation. MEXEC's planner then initiated the exchange of information with AutoNav to allow the planner to flesh out the template details to form a complete tasknet, which was then scheduled for execution. When the time came, MEXEC's controller executed the tasknet. This was successful, with AutoNav appropriately processing simulated imagery of three different geosynchronous targets over three successive orbits. The updated trajectory confidence was reported to MEXEC at the completion of the tasknet, and the goal of keeping orbit knowledge accuracy below the threshold was met for seven more orbits, at which time another optical navigation session was scheduled and executed, completely autonomously.

In summary, the following took place, all as expected, in the course of performing the demonstration:

- The tasknet templates were fully instantiated with
  - 13 states/timelines (e.g., for the predicted eclipse times, quaternions defining the spacecraft attitude, etc.).
  - 246 tasks scheduled (e.g., these included 36 *Take Image* tasks: 3 at each attitude x 4 attitudes per eclipse x 3 eclipses)
  - 16 task templates in uploaded XML (a template for each different element seen earlier in Figure 3: an overall *Improve Orbit Knowledge* template, and the items within it, e.g., *Image Session*, *Take Image*, etc.)
- MEXEC's execution of the tasknet conducted the following AutoNav activities over all three orbits:
  - Four attitude stations per orbit
  - Three images processed per attitude
  - One orbit solution per orbit
- Final *OD* (Orbit Determination) was performed with data from 36 (3 orbits x 3 spacecraft x 4 observations) images:
  - Updated eclipse times were pushed to MEXEC
  - Orbit knowledge updates occurred, advancing the projected time of the next accuracy threshold violation forward by 8 hours.
  - A follow-on *Improve Orbit Knowledge* session was planned and scheduled.

To see that the demonstration performed as expected, we inspected event logs, and scrutinized a useful visualization generated with output from the MEXEC planner. An example of the latter is in Figure 8, showing a view of several of the timelines after MEXEC has scheduled the *Improve Orbit Knowledge* tasknet.

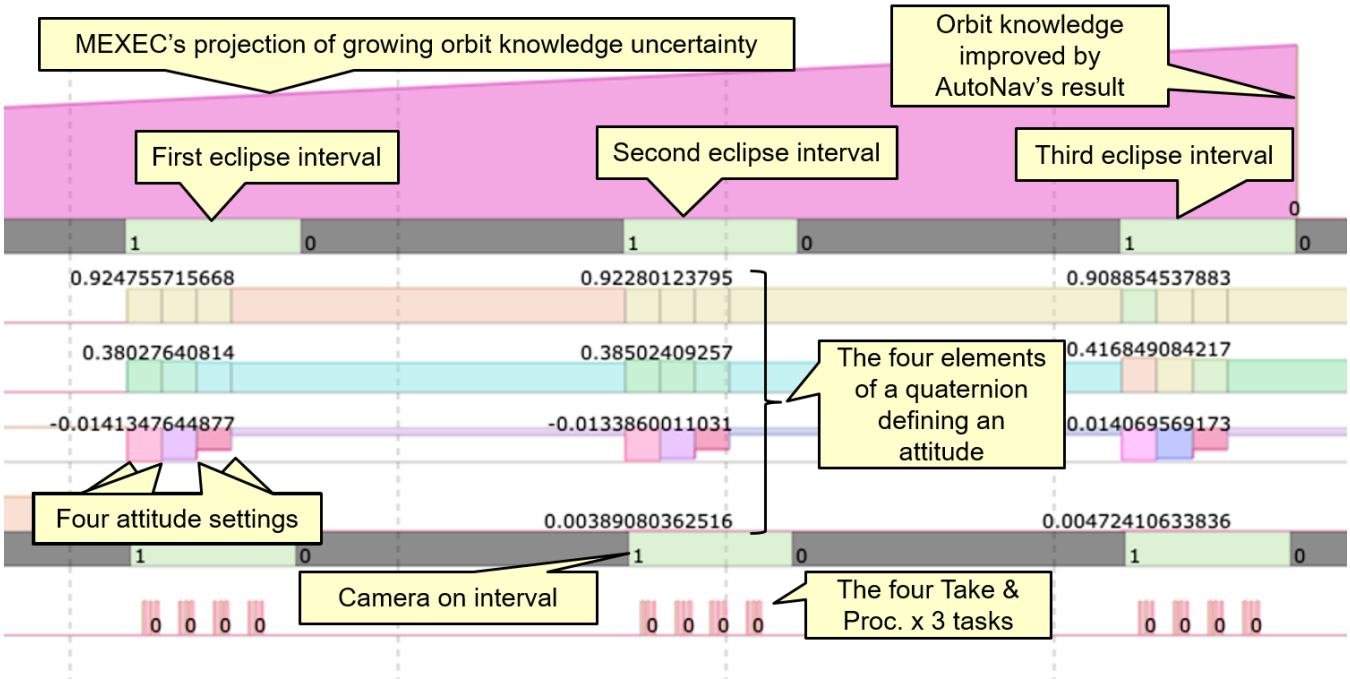


Figure 8 - Visualization of the scheduled tasknet

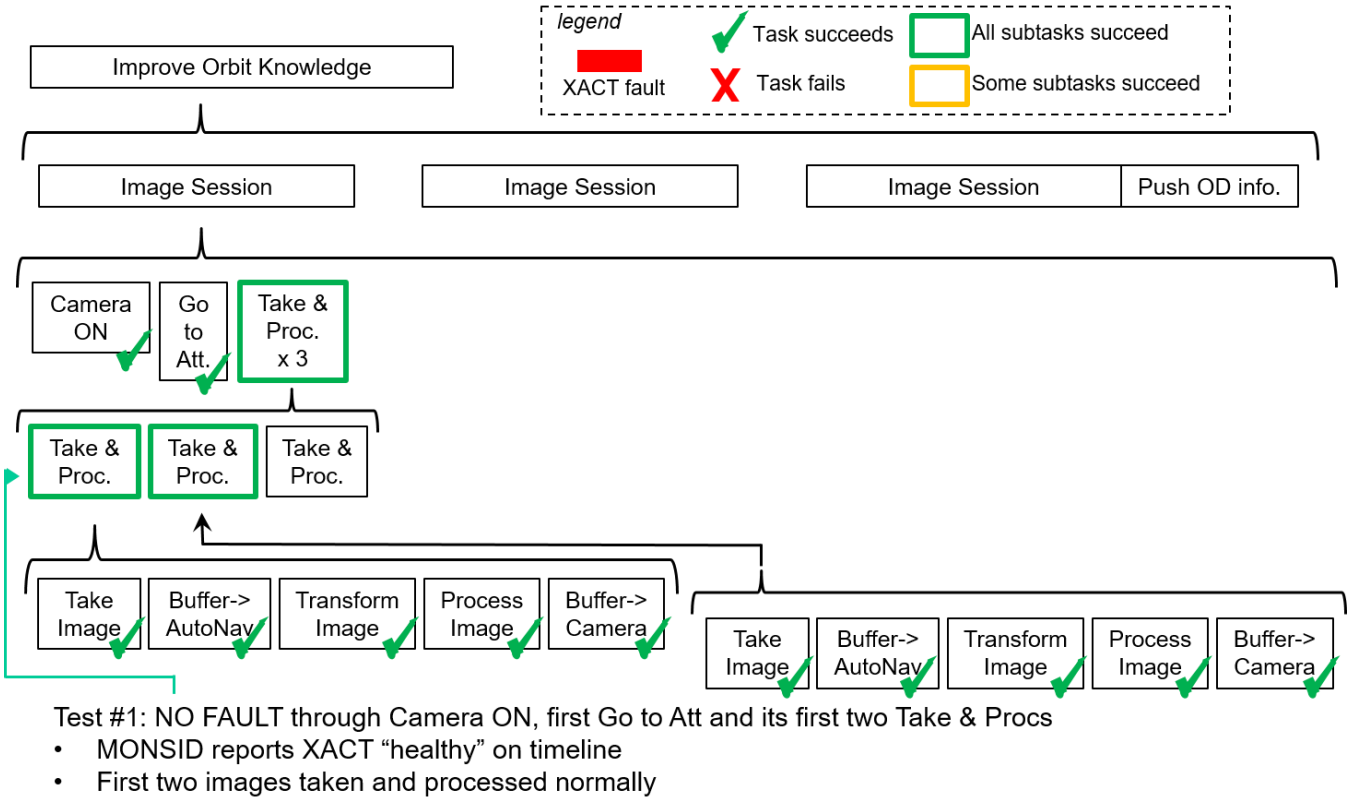


Figure 9 - Test #1: Nominal activity with MONSID actively monitoring and reporting healthy attitude control subsystem hardware

Nominal and Off-nominal Scenarios (with MONSID)

To enable off-nominal scenario testing, MONSID was integrated into the flight software with MEXEC and AutoNav, to monitor the health of the attitude control system. MONSID provided continuous status reports to the corresponding timeline in MEXEC via the state database.

Several tests were run with MONSID in place. Three of these tests are described next – a nominal test in which no fault was injected, a test of a fault injected at a time that would be inconsequential, and a test of a fault injected at a time that would be disruptive (along the lines of Figure 6). Because the scenario would take hours to run, these three tests were organized to occur in sequence within the same run.

Test #1: Nominal activity with MONSID actively monitoring—The scenario began the same way as did the nominal scenario, allowing everything to progress normally through the first two executions of the *Take Image* task in the first eclipse's *Image Session*. During this, the attitude control system performed normally. MONSID, fed data of the attitude control system's normal operation, continued to report it to be healthy. MEXEC successfully executed the two *Take Image* tasks, and the image manipulations that followed, as shown in Figure 9.

This affirmed that:

- The trio of MONSID, AutoNav and MEXEC were successfully integrated and able to execute together without exhausting computing resources.
- MONSID's communication to MEXEC's timeline of "healthy" reports was working.
- The tasknet and its execution by MEXEC correctly interpreted the "healthy" reports appearing on the timeline.

Test #2: An inconsequential fault injected—Following the third execution of the *Take Image* task in the first eclipse's *Image Session*, a transient fault was deliberately injected into the attitude control system during execution of AutoNav's image manipulation tasks (*Transform Image* and *Process Image*), *after* the image had been taken. The fault was made to be long enough that MONSID would recognize it and switch to reporting that the attitude control system was "unhealthy," but ended prior to the next *Go to Att.* in time for MONSID to switch back to reporting that that the attitude control system was "healthy". Because the *Transform Image* and *Process Image* tasks did not have constraints dependent on the attitude control system's health, MEXEC correctly allowed them to execute and succeed (Figure 10).

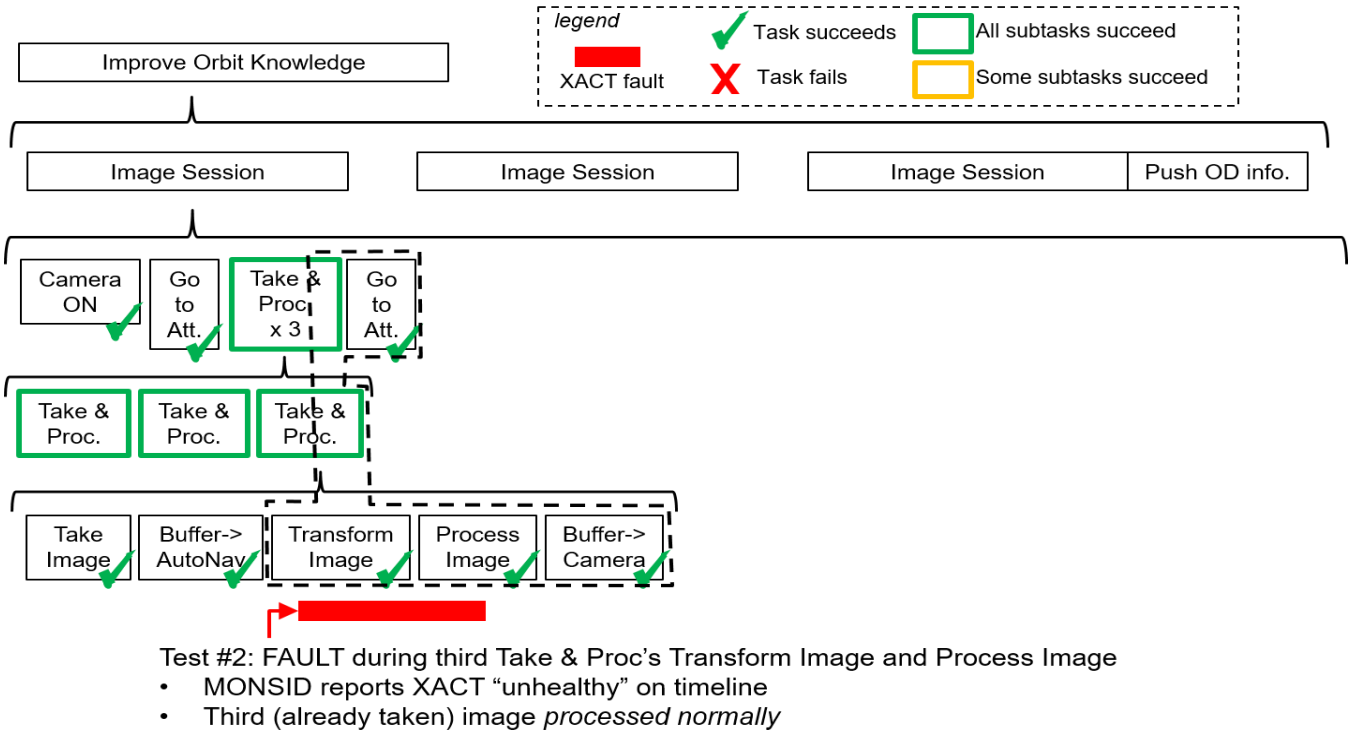


Figure 4 - Test #2: an inconsequential fault

This affirmed that:

- MONSID's communication to MEXEC's timeline of "unhealthy" reports was working.
- The tasknet and its execution by MEXEC was correctly unperturbed by "unhealthy" reports on timeline during execution of the *Transform Image* and *Process Image* tasks where healthy attitude control subsystem hardware is not required.

Test #3: A disruptive fault injected during Take Image— Next, the second *Go to Att.* and its first two *Take & Proc.* sessions were allowed to proceed normally – no faults were injected during this, and MEXEC correctly allowed the tasks to succeed. However, during the third *Take & Proc.*'s *Take Image* task, an attitude control system fault was injected in time for MONSID to recognize it and report "unhealthy" while the *Take Image* task's execution continued. Because this task had a maintenance constraint including the conjunct that *MONSID reports the attitude control system to be "healthy,"* MEXEC immediately failed the *Take Image* task. This was the intended behavior. While this simulated fault was transient in nature, allowing ACS to resume functioning without requiring a repair action, any interruption in fine point behavior can degrade the image collection or worse, and thus the transient fault resulted in failure of the *Take Image* task. Its failure led in turn to MEXEC not starting the *Buffer->AutoNav*, *Transform Image*, *Process Image*, and *Buffer->Camera* tasks, to avoid AutoNav processing a degraded image. The injected fault was ended, MONSID's detected this and reverted to reporting "healthy" before the next *Go to Att.*, and the three *Take Image* tasks that followed were seen to execute successfully, as described in Figure 11.

We saw this by examination of the detailed logs, aided by their rendering into a visualization of tasknet executions. Figure 12 shows a portion of this visualization of test #3. The green bar along the bottom displays MONSID's report of the attitude control system's health, with the short darker segment labelled "0" in the middle where MONSID had reported the transient fault. The red dashed lines were superimposed on the visualization to show where the fault report fell with respect to the Take Image task that was underway. That task was terminated by MEXEC, and the tasks to run the image processing steps that would normally have followed were cancelled by MEXEC. Note that whereas Figure 8 earlier was a visualization of the *scheduled* tasknet, Figure 12 is a visualization of tasknet *execution*.

This affirmed that:

- The tasknet and its execution by MEXEC was correctly recognizing and responding to violation of the *Take Image* task's maintenance constraint that XACT be reported as "healthy"
- The ripple effect of the *Take Image* task's failure was correctly occurring: task failures were correctly writing to internal states, and subsequent tasks' pre constraints were correctly accessing those internal states and interpreting them correctly
- Correct resumption of successful task executions occurred to allow the next *Go to Att.* to proceed.

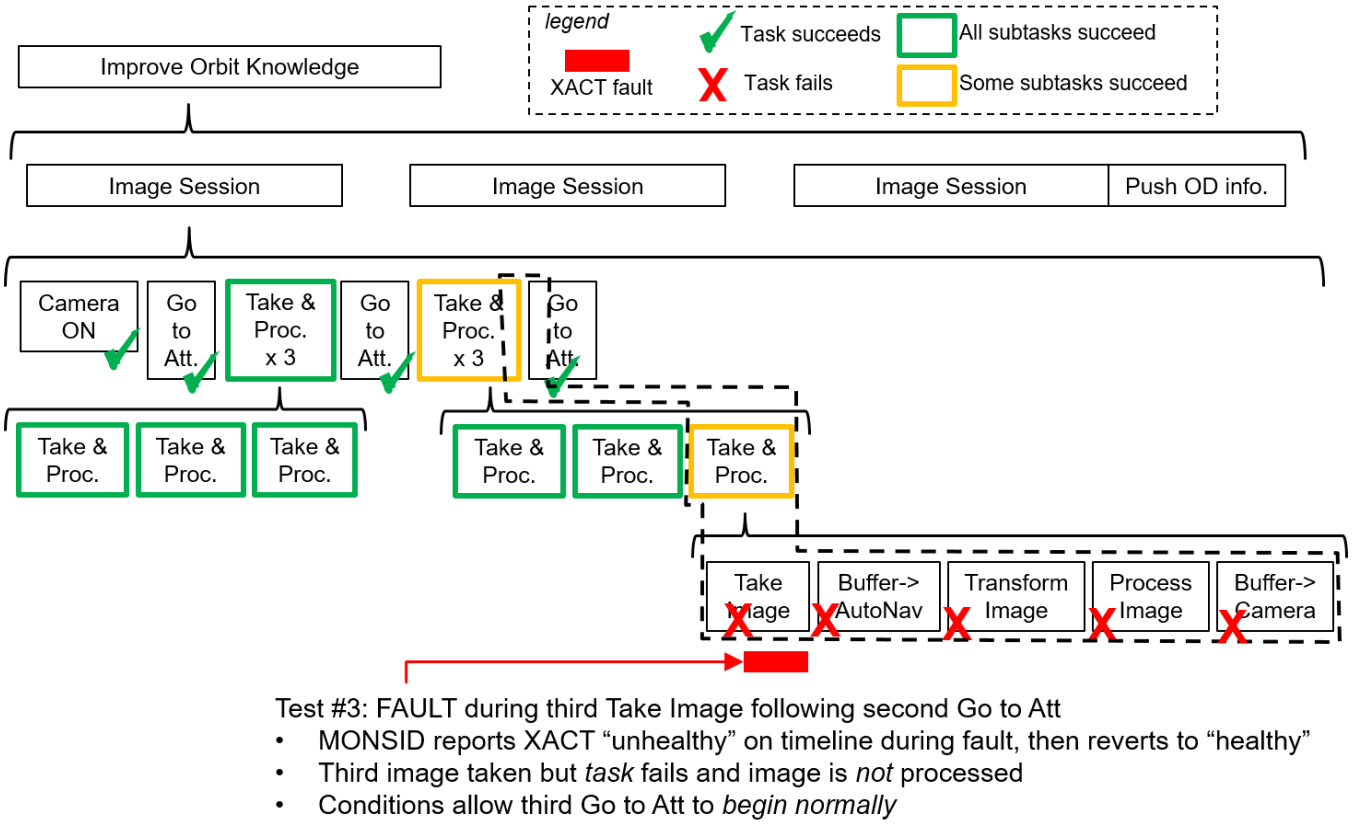


Figure 11- Test #3: A fault to the attitude control subsystem was injected during an observation, which disrupted the *Take Image* activity and subsequent activities that depend on a good image being taken

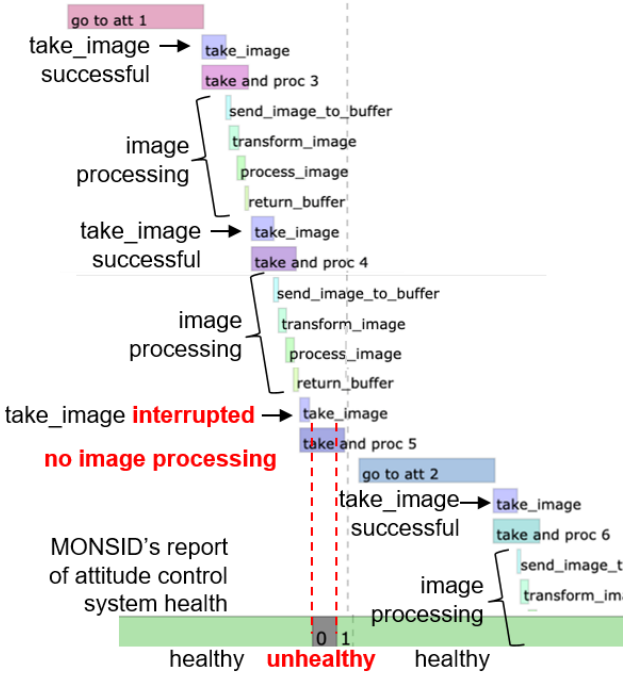


Figure 12 - Visualization of Test #3's tasknet execution

7. CONCLUSIONS

Scenario-specific implications

The scenario underpinning the demonstrations showed the feasibility of converting spacecraft orbit determination – what is traditionally a labor-intensive housekeeping activity repetitively performed by mission control – into a fully autonomous activity for a spacecraft to run by itself.

The work to make this possible was the integration of two of JPL's autonomy technologies together with health estimation technology. These were the AutoNav software, used previously to autonomously guide Deep Space 1's and Deep Impact's comet encounters, the MEXEC planner/executive software, used on ASTERIA in the summer of 2019 to control automated science data collection, and Okean Solution's MONSID software for monitoring the health of spacecraft hardware.

The demonstrations simulated an ASTERIA-like spacecraft in low Earth orbit. During the demonstration, MEXEC commanded the spacecraft to point and take pictures in the direction of other orbiting spacecraft whose locations were known. From identifying the position of those other spacecraft in the images, the AutoNav software accurately re-calculated the spacecraft's orbit. Faults injected into the attitude control system were recognized by MONSID, which then relayed health status information to MEXEC

(via a State Database), which in turn orchestrated the AutoNav activities to process only those images taken in the correct direction and held steady throughout image exposure.

The motivation for this scenario was that using the spacecraft for its scientific purpose requires accurate knowledge of its orbit. However, knowledge inaccuracies grow over time due to the small but cumulative effects of magnetic torque and gravity. In the demonstrations, MEXEC used an estimate of the inaccuracy growth rate to predict when it would exceed a threshold would make science impossible. To preempt this, MEXEC scheduled another session of AutoNav to recalculate orbit knowledge before that point. Once begun, this approach of MEXEC partnered with AutoNav would continue to keep orbit knowledge accurate enough for science, relieving mission control of this recurring housekeeping activity.

General implications

The successful demonstrations illustrated realization of the following general benefits of system-wide autonomy:

- **On-board planning, scheduling, and execution**
  - Planning to maintain goals in the future
  - Generation of conflict-free schedules
  - Execution of tasks in accordance with all constraints
- **Monitoring of and reaction to on-board system state**
  - Proactively scheduling *when needed*
  - Selection of an appropriate task to address impending conflicts
- **Hierarchical Tasks – decomposition and detailing methods**
  - Supporting the development and understanding of plans
  - Reduction of scheduling complexity
- **AutoNav as a function-level capability working with a system-level planner/executive subsystem**
  - Allowing the system-wide planner to call upon a domain-specific expert for estimation and calculations
- **Planner / domain-specific interface**
  - Enabling information exchange needed for on-board template instantiation to form tasks and tasknets for execution
- **Health monitoring**
  - Using a system perspective to recognize faulty components even when they may not know it themselves
  - Providing health monitoring estimates as another source of information available to the planner and execution engine
- **Health-informed execution**
  - Shielding autonomy's decisions and activities from untrustworthy data
  - Allowing unaffected decisions and activities continue unperturbed ("fail operational")
  - Resumption of normal operation when health restored

ACKNOWLEDGEMENTS

This research was carried out at Okean Solutions, Inc. and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004) and funded through the internal Research and Technology Development program.

REFERENCES

[1] Fesq, L. et al, "Results from the ASTERIA CubeSat Technology Demonstrations." To be published in *IEEE Aerospace Conference Proceedings*, March 2021.

[2] 2018 Workshop on Autonomy for Future NASA Science Missions: Output and Results. Science Mission Directorate. https://science.nasa.gov/technology/2018-autonomy-workshop/output-results

[3] Amini et al, "Advancing the Scientific Frontier with Increasingly Autonomous Systems." Planetary Science and Astrobiology Decadal 2023-2032 White Paper.

[4] Bernard, D.E., Dorais, G.A., Fry, C., Gamble, E.B., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P.P., Pell, B. and Rajan, K., "Design of the remote agent experiment for spacecraft autonomy." *1998 IEEE Aerospace Conference Proceedings* (Cat. No. 98TH8339) (Vol. 2, pp. 259-281). IEEE, March 1998.

[5] Rajan, K., Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P., Rouquette, N. and Smith, B., "Remote Agent: An autonomous control system for the new millennium." In *Proceedings of the 14th European Conference on Artificial Intelligence* (pp. 726-730). IOS Press, August 2000.

[6] Muscettola, N., "HSTS: Integrating planning and scheduling" (No. CMU/RI-TR-93-05). CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST. 1993

[7] Muscettola, N., Fry, C., Rajan, K., Smith, B., Chien, S., Rabideau, G. and Yan, D., "On-board planning for new millennium deep space one autonomy." In *1997 IEEE Aerospace Conference* (Vol. 1, pp. 303-318). IEEE. February, 1997.

[8] Williams, B.C. and Nayak, P.P., "A model-based approach to reactive self-configuring systems." In *Proceedings of the national conference on artificial intelligence* (pp. 971-978). August 1996.

[9] Pell, B., Gat, E., Keesing, R., Muscettola, N. and Smith, B., 1997, "Robust periodic planning and execution for autonomous spacecraft." In *IJCAI* (pp. 1234-1239). August 1997.

[10] Nayak, P., Kurien, J., Dorais, G., Millar, W., Rajan, K., Kanefsky, R., Bernard, E.D., Gamble Jr, B.E., Rouquette, N., Smith, D.B. and Tung, Y.W., "Validating the ds-1 remote agent experiment." In *Artificial intelligence, robotics and automation in space* (Vol. 440, p. 349). August 1999.

[11] Amini, R., Fesq, L., Kolcio, K., Mackey, R., Mirza, F., Rasmussen, R., Troesch, M., "FRESCO: A Framework for Spacecraft Systems Autonomy." To be published in *IEEE Aerospace Conference Proceedings*, March 2021.

[12] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davis, A., Mandl, D., Frye, S., Trout, B. and Shulman, S., "Using autonomy flight software to improve science return on Earth Observing One." Journal of Aerospace Computing, Information, and Communication, 2(4), pp.196-216, 2005.

[13] Knight, S., Rabideau, G., Chien, S., Engelhardt, B. and Sherwood, R., "Casper: Space exploration through continuous planning." IEEE Intelligent Systems, 16(5), pp.70-75. 2001.

[14] Tipaldi, M. and Glielmo, L., "A survey on model-based mission planning and execution for autonomous spacecraft." *IEEE Systems Journal*, 12(4), pp.3893-3905, 2017.

[15] Cividanes, F., Ferreira, M. and Kucinskis, F., "On-board Automated Mission Planning for Spacecraft Autonomy: A Survey." *IEEE Latin America Transactions*, 17(06), pp.884-896, 2019.

[16] Bauer, B.A. and Reid, W.M., "Automating the Pluto Experience: An Examination of the New Horizons Autonomous Operations Subsystem." In *2007 IEEE Aerospace Conference* (pp. 1-10). IEEE, March 2007.

[17] Steiger, C., Pasetti, A., Espeillac, P., Strandberg, T., Casasco, M., Altay, A. and Montagnon, E., "Autonomy and FDIR Challenges for the BepiColombo Mission To Mercury." *GNC* 2017.

[18] Stottler, D., Ramachandran, S., Belardi, C. and Mandayam, R., "On-board, autonomous, hybrid spacecraft subsystem fault and anomaly detection, diagnosis, and recovery." *Advanced Maui Optical and Space Surveillance Technologies Conference* (AMOS), 2020.

[19] Wander, A. and Förstner, R., "Innovative fault detection, isolation and recovery on-board spacecraft: Study and implementation using cognitive automation." In *2013 Conference on Control and Fault-Tolerant Systems* (SysTol) (pp. 336-341). IEEE, October 2013.

[20] Schmidt, M., "The ECSS Standard on Space Segment Operability." In *Space OPS 2004 Conference* (p. 504), May 2004.

[21] Johnson, A., Wilson, R., Cheng, Y., Goguen, J., Leger, C., San Martin, M. and Matthies, L.,"Design Through Operation of an Image-Based Velocity Estimation System for Mars Landing." *International Journal of Computer Vision* 74(3), pp 319–341, 2007.

[22] Rankin, A., Maimone, M., Biesiadecki, J., Patel, N., Levine, D. and Toupet, O., "Driving Curiosity: Mars Rover Mobility Trends During the First Seven Years." In *2020 IEEE Aerospace Conference* (pp. 1-19). IEEE. March 2020.

[23] Francis, R., Estlin, T., Johnstone, S., Peret, L., Mousset, V., Doran, G., Gaines, D., Montaño, S., Gasnault, O., Frydenvang, J. and Wiens, R., "Incorporating AEGIS autonomous science into Mars Science Laboratory rover mission operations." *2018 SpaceOps Conference*, p 2576, 2018.

[24] Bhaskaran, S., "Autonomous Navigation for Deep Space Missions." *Proceedings of the SpaceOps 2012 Conference*, AIAA Paper 2012- 1267135, 2012.

[25] Smith, M., Donner, A., Knapp, M., Pong, C., Smith, C., Luu, J., Pasquale, P.D. and Campuzano, B., 2018. "On-orbit results and lessons learned from the ASTERIA space telescope mission." *32rd Annual AIAA/USU Conference on Small Satellites*, 2018.

[26] Fesq, L., Beauchamp, P., Donner, A., Bocchino, R., Kennedy, B., Mirza, F., Mohan, S., Sternberg, D., Smith, M.W., Troesch, M. and Knapp, M., "Extended Mission Technology Demonstrations Using the ASTERIA Spacecraft." In *2019 IEEE Aerospace Conference* (pp. 1-11). March 2019.

[27] Bocchino, R., Canham, T., Watney, G., Reder, L. and Levison, J., "F Prime: An Open-Source Framework for Small-Scale Flight Software Systems." *Proc. AIAA/USU Conference on Small Satellites*, 2018.

[28] Bhaskaran, S., Riedel, J., Synnott, S. and Wang, T., "The Deep Space 1 autonomous navigation system-A post-flight analysis." *In Astrodynamics Specialist Conference* (p. 3935). August 2000.

[29] Bhaskaran, S., "Autonomous navigation for deep space missions." In *SpaceOps 2012* (p. 1267135). 2012.

[30] Kennedy, B.M., Doran, P., Hughes, R.S., Hughes, K., Bocchino, R., Lubey, D., Mages, D. and Fesq, L., "Satellite-to-satellite imaging in support of LEO optical navigation, using the ASTERIA CubeSat." *In CubeSats and SmallSats for Remote Sensing IV* (Vol. 11505, p.

115050H). International Society for Optics and Photonics. August 2020.

[31] Kolcio, K. and Fesq, L., "Model-based Off-nominal State Isolation and Detection System for Autonomous Fault Management," in *2016 IEEE Aerospace Conference*, pp. 1-13, IEEE, 2016.

[32] Davis, R., "Diagnostic Reasoning Based On Structure And Behavior," *Artificial intelligence* 24, no. 1-3 (1984): pp. 347-410.

[33] Kolcio, K., Mackey, R. and Fesq, L., "Model-Based Approach to Rover Health Assessment-Mars Yard Discoveries." In *2019 IEEE Aerospace Conference* (pp. 1-12). IEEE. March 2019.

[34] Nikora, A., Srivastava, P., Fesq, L., Chung, S. and Kolcio, K., "Assurance of model-based fault diagnosis." In *2018 IEEE Aerospace Conference* (pp. 1-14). IEEE. March 2018.

[35] Mackey, R., Altenbuchner, C., Sievers, M., Nikora, A., Fesq, L., Kolcio, K., Prather, M. and Litke, M., "On-Board Model Based Fault Diagnosis for Cubesat Attitude Control Subsystem: Flight Data Results." To be published in *IEEE Aerospace Conference Proceedings*, March 2021.

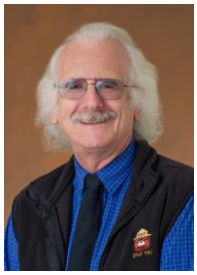
[36] Verma, V., Gaines, D., Rabideau, G., Schaffer, S. and Joshi, R., "Autonomous Science Restart for the Planned Europa Mission with Lightweight Planning and Execution," *International Workshop on Planning and Scheduling for Space* (IWPSS 2017), Pittsburgh, PA 2017.

[37] Biesiadecki, J.J., Henriques, D.A. and Jain, A., "A reusable, real-time spacecraft dynamics simulator." In *16th DASC. AIAA/IEEE Digital Avionics Systems Conference. Reflections to the Future*. Proceedings (Vol. 2, pp. 8-2). IEEE. October 1997.

[38] Pong, C. M., Sternberg, D. C. and Chen, G. T., "Adaptations of guidance, navigation and control verification and validation philosophies for small spacecraft." In *Proceedings of the 42nd Annual AAS Rocky Mountain Section Guidance and Control Conference*, Breckenridge, Colorado, 2019.

[39] Troesch, M., Mirza, F., Hughes, K., Roshsetin-Dowden, A., Bocchino, R., Donner, A., Feather, M,. Smith, B., Fesq, L., Barker, B. and Campuzano, B., "MEXEC: An Onboard Integrated Planning and Execution Approach for Spacecraft Commanding." *Workshop on Integrated Execution (IntEx) / Goal Reasoning (GR), International Conference on Automated Planning and Scheduling* (ICAPS IntEx/GR 2020), October 2020.

BIOGRAPHY



Martin Feather *is a Principal Software Assurance Engineer in JPL's Office of Safety and Mission Success, and has been with JPL for 25 years. His activities are focused on identifying, developing and infusing research results that offer the prospect of improving assurance of space missions, with a particular interest in space systems' software. He is the recipient of a NASA Exceptional Achievement Medal and has published over 150 papers. He received his BA and MA in Mathematics and Computer Science from the Cambridge University, UK, and PhD in Artificial Intelligence from the University of Edinburgh, UK, all in the previous millennium.*

For the past twenty years, ***Brian Kennedy*** *has been a member of the Mission Design and Navigation section at NASA's Jet Propulsion Laboratory. Brian is currently the lead for the ASTERIA Autonomous Optical Navigation SR&TD team. He was until recently the Orbit Determination Team Lead for the Dawn Mission and the Lead for the OSIRIS-REx Independent Bennu Asteroid Shape Model Team. His past work at JPL has included support for the Deep Space 1, Odyssey, MER, Stardust, Deep Impact, LCROSS and EPOXI missions. Brian graduated from The University of Colorado in 1993 with a B.S. in Aerospace Engineering Sciences.*

Ryan Mackey *is a senior software systems engineer and researcher of Integrated System Health Management technologies at JPL. He is a graduate of the Graduate Aeronautical Laboratories, California Institute of Technology (GALCIT). His projects focus on development of anomaly detection, data mining, and prognostics technologies along with applications to air and space systems and cybersecurity His current tasks include system engineering for the Europa Clipper mission and application of Integrated System Health Management technologies to aircraft and spacecraft. He has been granted three US Patents for his contributions o ISHM.*

Martina Troesch *is a member of the Artificial Intelligence group at the Jet Propulsion Laboratory. She holds a B.S. and M.S. from the University of Southern California in Aerospace Engineering, as well as an M.S. in Computer Science from Stanford University.*

Cornelia Altenbuchner *is currently a Robotics Technologist at the NASA Jet Propulsion Laboratory (JPL) in Pasadena California. Cornelia earned her PhD from the University of Maryland College Park in Aerospace Engineering, during which time she conducted research at the NASA Langley Research Center and the National Institute of Aerospace. Her primary contributions are technology development in the areas of flexible multi-body dynamics modeling and simulation, robotic systems, conceptual mission design, dynamics and controls as well as Flight Software development. At NASA JPL, she works on robotic systems, which includes dynamic, autonomy and conceptual aspects required to support missions to Europa and Mars 2020.*

Robert Bocchino *is a Flight Software Engineer in the Small-Scale Flight Software Group at JPL. He holds a PhD in computer science from the University of Illinois at Urbana-Champaign, and he was a Postdoctoral Associate at Carnegie Mellon University. Since joining JPL in 2013, Robert has worked on both technology development and flight projects. He was the flight software technical lead for the ASTERIA missions. Robert is currently working on technology development in the areas of software development and testing, autonomous flight software, and high-performance space computing.*

Patrick Doran *is currently an employee of Pearl River Technologies, LLC and is in the process of completing his Master's of Science in Aerospace Engineering at California State Polytechnic University, Pomona. His University research has been focused on autonomous swarm satellite navigation. Previously, he spent two years interning for the Mission Design and Navigation section at NASA's Jet Propulsion Laboratory, where he supported the ASTERIA Autonomous Optical Navigation SR&TD team.*

.

Lorraine Fesq *is a Principal Engineer at the Jet Propulsion Laboratory, California Institute of Technology, and was the Project Manager of the ASTERIA mission. She serves as Associate Director for the Caltech Center for Autonomous Systems and Technologies. Lorraine has over 30 years of aerospace experience that spans industry, government and academia, and has worked all mission phases of spacecraft development. Lorraine holds two patents and has received a NASA Public Service Medal for her work on the Chandra X-ray Observatory and a NASA Exceptional Achievement Medal for advancing the Fault Management discipline within NASA. Lorraine taught in the Aeronautics/Astronautics department at MIT and worked on multiple spacecraft projects at TRW and Ball Aerospace. She received her B.A. in Mathematics from Rutgers University and her M.S. and Ph.D. in Computer Science from the University of California, Los Angeles.*

Randall Scott Hughes *is an Engineering Systems Software Engineer in the Mission Design and Navigation software group at the Jet Propulsion Laboratory. Since joining JPL in 2017 he has worked on several projects including cubesat hardware testing, orbit determination for the MAVEN Mars orbiter, development of asteroid shape modeling software, cubesat flight software development for the ASTERIA mission, and development of JPLs Monte navigation software. He holds a Ph.D. in Astronautical Engineering from the University of Southern California with a focus on large-scale simulations of kinetic turbulence in the solar wind. While at USC he also received an M.S. degree in Astronautical Engineering and an M.S. degree in Computer Science with a focus on high performance computing and scientific simulation. He received his B.S. in Aerospace Engineering from UC San Diego.*

Faiz Mirza *is a Member of Technical Staff in the Artificial Intelligence Group (397I), Planning and Execution Section, of the Jet Propulsion Laboratory, California Institute of Technology. Faiz graduated from the University of Edinburgh with a M.S. in Artificial Intelligence. He was an undergraduate at the University of California - Riverside, where he completed a B.S. in Computer Science. Previously he was an intern at Amazon. While at Edinburgh, he was a Teaching Assistant for a Software Testing course and while at UC Riverside he worked as a systems administrator.*

Allen Nikora *is a Principal Software Assurance Engineer in JPL's Quality Assurance Office, and has been with the Jet Propulsion Laboratory, California Institute of Technology for over 40 years. With support from NASA's Office of Safety and Mission Assurance, the U. S. Air Force Operational Test and Evaluation Center, the U. S. Naval Air Systems Command, and JPL internal funding, he has conducted extensive research in various aspects of software reliability engineering and defect analysis, and has developed tools for estimating software reliability*

growth during test and operations. He has published extensively on his work, and has been active in the IEEE standards community, participating as a working group member and chair for a number of software reliability-related IEEE standards including all IEEE Std 982.1-2005 and versions of IEEE Std 1633. He received the NASA Exceptional Achievement award for contributions to the successful December, 1995 GALILEO entry into Jupiter orbit and relay of probe data and the NASA Inventions and Contributions Award and Space Act Award for the development of the CASRE software reliability tool. He holds a B.S. in Engineering and Applied Science from the California Institute of Technology, and a Ph.D. in Computer Science from the University of Southern California. He belongs to the IEEE, the IEEE Computer Society, and the IEEE Reliability Society.

.

Ksenia Kolcio *is the Vice President of Okean Solutions, an SBC located in Seattle, WA, where she leads aerospace systems engineering technical activities. Ksenia has been the PI on several SBIR Phase I, II, and III programs with the Air Force Research Laboratory and NASA JPL focusing on model-based fault management solutions that strive to increase spacecraft autonomy. Ksenia has also subcontracted with partner companies on other NASA Phase II and III programs, and DoD BAA contracts in the areas of spacecraft attitude control and navigation mission analysis. Prior to co-founding Okean Solutions, she was employed at Microcosm, Inc. for seven years as a Senior Systems/GN&C Engineer where she led and managed GN&C hardware & software development, CONOPS, and systems analysis on SBIR Phase I, II, and III programs with NASA and DoD. Prior to Microcosm, she worked at Northrop Grumman Space Technology (NGST) for 8 years in the Avionics Systems Center. At Northrop Grumman Ksenia worked on a variety of NASA and DoD flight programs from design to Integration and Test. In addition to spacecraft control design, analysis, and test she was an autonomy and fault management lead engineer responsible for fault management architecture development and testing. She received a Ph.D. in Electrical Engineering from University of Cincinnati. She is a member of AIAA.*

Matthew Litke *is the Directory of Systems Engineering at Okean Solutions, Inc. He has been finding engineering solutions to integrate hardware and software for over 20 years. He is currently working on integrating the MONSID fault management system with several spacecraft architectures and frameworks. He received his B.S. in Aerospace Engineering and Mechanics from University of Minnesota and his M.S. Mechanical Engineering from Stanford University.*

Maurice Prather *is the co-owner of Okean Solutions, Inc. and currently serves as the company President. Maurice received a Bachelors in Aerospace Engineering and Masters in Mechanical Engineering from the University of Alabama. Maurice has spent nearly 22 years in the aerospace and software industries. He currently works as a system architect, developer and trainer, including lead development architect/manager for the MONSID fault management system. Maurice started his career as an aerospace engineer working on various software programs in Boeing's commercial airplane division in the noise and flight operations areas. Afterwards, Maurice worked for Microsoft as a software design engineer in test working on one of Microsoft's most popular business product lines. For the past 11 years, Maurice has worked as an IT consultant providing architectural guidance, leading development teams, and building custom applications for a large number of corporate and government clients.*

Patricia M. Beauchamp *is the Chief Technologist for the Engineering and Science Directorate at the Jet Propulsion and also a founding member of Planetary Exploration Science Technology Office (PESTO) within the NASA PSD. She was the deputy PI of a Venus Flagship Mission Concept Study for the Planetary Decadal Survey. She has been on the executive committees of the Outer Planet and Venus Science Assessment Groups and has been responsible and/or involved in developing technology plans for the Outer Planets, Venus and Small Bodies Assessment Groups. Her previous JPL roles include serving as Manager of the Planetary Instrument Development Office, Leader of the Center for In-Situ Exploration and Sample Return (CISSR), and Project Manager for the MICAS Instrument on DS1. She is the recipient of the multiple Aerojet, JPL and NASA awards. Pat received her PhD in Chemistry from Caltech in 1981 followed by a post-doctoral fellowship in Chemical Engineering. Before joining JPL in 1991 she managed a detector and materials research department at Aerojet ElectroSystems.*