Autonomous Science Restart for the Planned Europa Mission with Lightweight Planning and Execution

Vandi Verma, Dan Gaines, Gregg Rabideau, Steve Schaffer, Rajeev Joshi
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109
{*firstname.lastname*}@jpl.nasa.gov

Abstract

In this paper we present MEXEC - a lightweight on-board planning and execution system that monitors spacecraft state to robustly respond to current conditions. In addition it projects the remaining plan forward in time to detect conflicts and revise the plan.

Most current planetary missions use sequence based commanding from the ground, which limits the ability to respond to on-board state varying from planned for state. This results in planning for worst-case execution time, power utilization, data volume and other resources and leads to underutilization of the spacecraft capability. It also limits the ability to respond to faults.

Due to the high radiation environment at Jupiter the probability of flight software resets is high. Therefore, of particular interest to the planned Europa Mission is the capability to restart the science plan after flight software resets. In this paper we present results from running Europa flyby scenarios with flight software resets in the Europa flight software test environment. Preliminary results with the prototype scenario show MEXEC takes less than a tenth of a second to respond to resets. The development was done as part of Europa-focused flight software, but MEXEC was designed to be applicable to landers and rovers as well.

Introduction

The Europa Clipper mission is baselined to orbit Jupiter and conduct 45 low-altitude flybys of its moon Europa. Each flyby would be approximately 10-20 hrs as part of an orbit of approximately 14 days. The duration of flyby and orbit would vary for each elliptical orbit. The radiation the spacecraft would experience is high and this is expected to cause the flight software to reset. The current requirement is to accommodate five flight software resets during the Europa flyby segment. The highest priority science for the mission occurs during the flybys. There is a requirement to autonomously recover science activities after a reset since there is insufficient time to transmit spacecraft state and receive an updated plan from Earth. Figure 1 shows a simplified example of the spacecraft orbit around Jupiter and flyby of Europa. Each petal is expected to be slightly different. Information about the Europa mission is available at `https://www.jpl.nasa.gov/missions/europa-clipper/`.

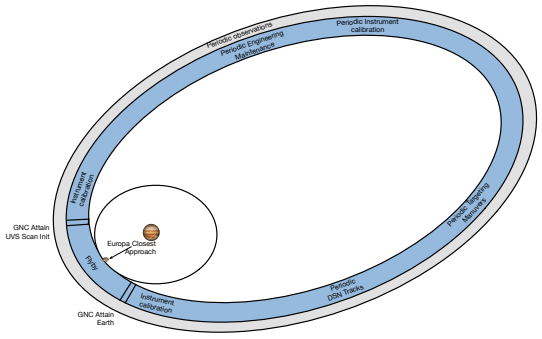

Figure 1: Simplified Jovian Orbit and Flyby Segment.

Sequencing Approach and Related Work

Existing deep space spacecraft such as Cassini, Mars Exploration Rovers, and Curiosity use a sequencing based approach for commanding. Science and engineering intent and constraints are taken into consideration during ground based planning. Based on the intent and constraints a set of time and event driven activities are generated. A simple example is shown in Figure 2. The intent in the example is to collect Ultraviolet Spectrograph (UVS) data with the instrument slewing across Europa. In order to slew the instrument the entire spacecraft must be slewed. Hence the spacecraft Guidance and Navigation Controller (GNC) must attain the initial attitude to start the scan, then command UVS to collect data while co-ordinating GNC to slew for the UVS scan. The UVS instrument temperature must be maintained within operating range while it is scanning. To reach the operating temperature the thermal system needs to be commanded to turn on the heaters. The heaters take a period of time to reach the operating range, hence the warmup activity must be scheduled accordingly to ensure the instrument is at operating range by the time GNC has achieved the initial scan attitude. There are many additional constraints and impacts that are not shown in the simplified example. For example the impact on power from turning on the heaters and the in-

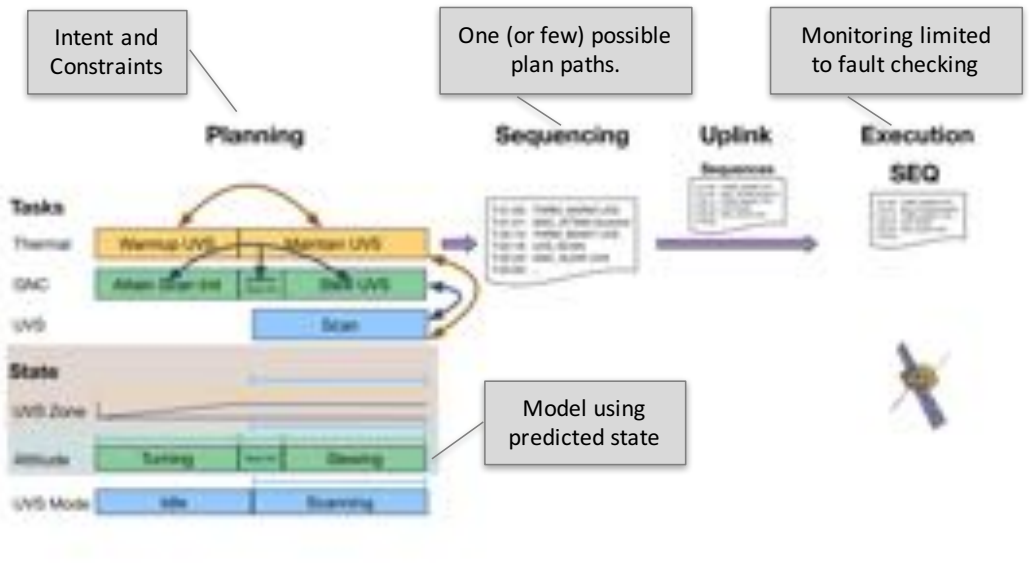

Figure 2: Existing sequencing approach

strument.

Ground tools are used to expand activity templates, model constraints and impact on state, and detect conflicts. Activities are updated based on the feedback until a conflict free plan is generate on the ground. Once a conflict free ground based plan is generated, it is translated into sequences. Sequences capture one (or a few) possible plan paths. Once the plan is converted to sequences much of the original science and engineering intent and constraints are lost. The sequences that implicitly meet the intent and constraint given predicted state are uplinked to the spacecraft. On-board monitoring is typically limited to fault checking. To mitigate faults typically considerable margin is maintained in the plan, which results in under-utilization of resources. However, they cannot completely be avoided such as in the event the spacecraft encounters a flight software reset.

MEXEC Approach

In the MEXEC approach shown in Figure 3 , the intent and constraints available during ground based planning are preserved and uplinked to the spacecraft. MEXEC flight software executes the plan. It monitors on-board state to robustly respond to actual state and uses current state to perform on-board constraint checking.

The ability to robustly respond to actual state and perform on-board constraint checking for the executing and remaining plan enables fail operational capability, such as after a flight software reset. In addition, it simplifies uplink product creation and review since intent and constraints are explicitly captured and checked on-board. This reduces daily operations tactical planning overhead for command product generation.

The MEXEC approach was designed to leverage existing sequencing capability. A MEXEC task can represent a sequence. This simplifies sequence creation by decoupling sub-system interactions from sequencing. Sub-system interactions, such as the dependency of the UVS sub-system on the GNC and Thermal sub-systems as shown in Figure 2, are managed via constraints. The flexible choice between sequencing and commanding with higher level intent allows incremental improvement over sequencing.

Constraints and Impacts

To perform lightweight on-board constraint checking for the executing and remainder of the plan, each task in an MEXEC plan includes constraints and impacts on resources.

- A **resource** is a value over time. For example, time/duration, data volume, energy, claims, states.
- A **constraint** is a requirement on a resource value at a specific time, or over a time interval. For example power spikes must stay below 100W, data volume is empty at start of flyby, temperature between 10 and 20 degrees during observation, claim on an instrument during observation, attitude must be nadir during observation. A *PRE CONSTRAINT* must be satisfied just before the start of the task. A *MAINTAIN CONSTRAINT* must be satisfied from (and including) the task start and up to the task end.
- An **impact** is a change in a resource value at a specific time, or over a time interval. For example, power load of an observation, data volume consumed by an observation, temperature change from a heating activity, claim on an instrument during an observation, slew changes pointing state. A *PRE IMPACT* occurs at the start of the task, a *MAINTAIN IMPACT* occurs at the start of the task and up to the task end, a *POST IMPACT* occurs at the end of the task.

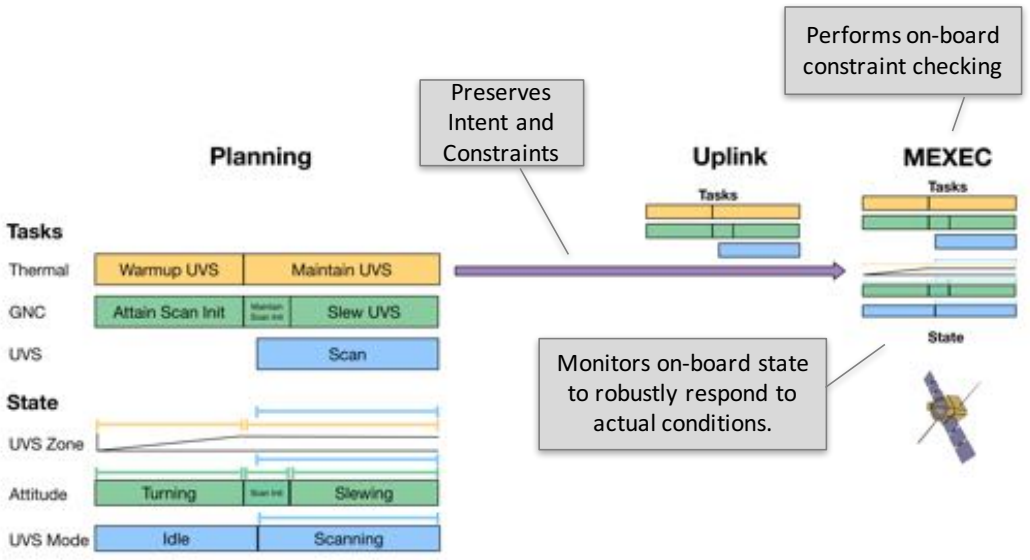

Figure 3: MEXEC approach.

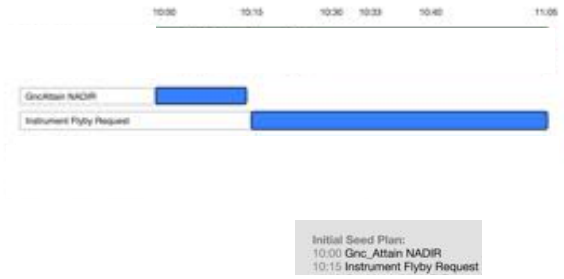

Figure 4: Nominal MEXEC plan represented in Gantt chart form.

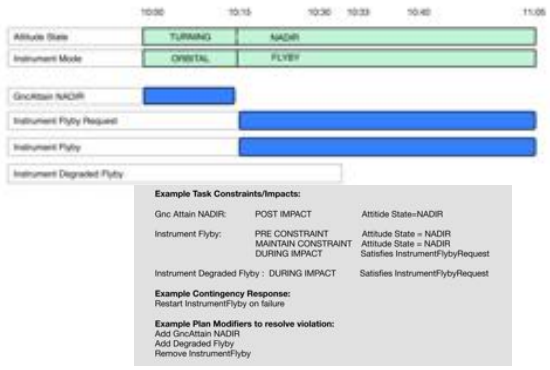

Figure 5: Example constraints and impacts.

Example Flyby Scenario

Figure 4 shows an example of nominal MEXEC plan that captures intent in Gantt chart form. Planned tasks are shown in blue. The time notation has been simplified for discussion. It shows that *GncAttain NADIR* task is scheduled to start at 2028-040T10:00 for a duration of 15min and an *Instrument Flyby Request* is scheduled for 2028-040T10:15 for a duration of 50min. Figure 5 shows an expanded view of the plan. The *Instrument Flyby* task is scheduled to satisfy the *Instrument Flyby Request*. The green bars on the top shows examples of spacecraft state. Europa flight software has a State Buffer Store (SBS) module that provides state updates to all flight software modules including MEXEC. Figure 5 provides examples of task constraints and impacts as well. For example:

- The *GncAttain NADIR* task has the *POST IMPACT* after task completion that the *Attitude State* is *NADIR*.
- The *Instrument Flyby* task has the *PRE CONSTRAINT* that it requires the *Attitude State* to be *NADIR* at the start.
- The *Instrument Flyby* task also has a *MAINTAIN CONSTRAINT* requiring the *Attitude State* to be *NADIR* for the duration of task execution.
- The *Instrument Flyby* task has a *DURING IMPACT* that for the duration of the task it satisfies the *Instrument Flyby Request*.
- The *Instrument Degraded Flyby* task also has the *DURING IMPACT* that for the duration of the task it satisfies the *Instrument Flyby Request*.

Example Flyby Reset Scenario

Figure 6 shows nominal plan execution until 2028-040T10:30 at which time the spacecraft encounters a flight software reset. Blue boxes show the initial plan and the orange lines show the as-executed plan. The Europa spacecraft is baseline to take three minutes to run fault protection and

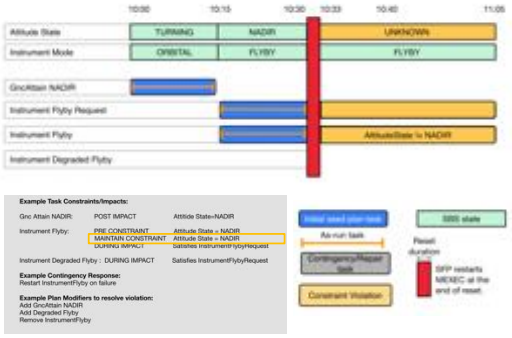

Figure 6: Example constraint violation due to flight software reset.

recover from the fight software reset. On recovery, system fault protection restarts MEXEC. MEXEC restores the task network and plan state. It queries current spacecraft state and determines that the *Attitude State* is *UNKNOWN*. This violates the *MAINTAIN CONSTRAINT* for the *Instrument Flyby* task which requires the attitude state to be *NADIR*. Due to the violation of the *MAINTAIN CONSTRAIN* constraint the *Instrument Flyby* task is cleanly aborted. This is done at the MEXEC controller level (described in the MEXEC Planner and Controller section) and the MEXEC planner is notified.

As part of MEXEC development we interviewed stakeholders from a wide variety of past and existing deep space missions, including orbiters, landers and rovers. One of the features that many desired was the ability to explicitly specify contingency responses for a given situation. MEXEC allows contingency responses to be conditioned on failure type. Figure 7 shows an example of an explicit contingency response for the *Instrument Flyby* task. On failure the contingency response was to restart the task.

To schedule the *Instrument Flyby* task MEXEC still ensures that all the task constraints and impacts for the task and the remainder of the plan are considered. It detects that the *PRE CONSTRAINT* for the *Instrument Flyby* task requires that the *Attitude State* be *NADIR*.

MEXEC allows specification of plan modifiers to resolve violations. Examples, of plan modifiers include adding, deleting, and moving tasks. Based on feedback from stakeholders MEXEC provides a mechanism to explicitly specify permissions for modifying specific tasks. Figure 7 shows that in this scenario plan modifiers included adding *GncAttain NADIR* and *Instrument Degraded Flyby*. When MEXEC detects the *PRE CONSTRAINT* violation when scheduling the *Instrument Flyby* task it resolves it by adding a *GncAttain NADIR* task before it since it has a *POST IMPACT* of changing the *Attitude State* to *NADIR*. Then there remains a violation that the *Instrument Flyby Request* is no longer satisfied for the period that the spacecraft is attaining NADIR. To resolve this MEXEC adds a *Instrument Degraded Flyby* task that does not have a *PRE CONSTRAINT* for *NADIR* pointing and collects instrument data in a degraded mode.

Figure 8 shows that unlike sequencing approaches the same MEXEC plan can handle a wide variety of reset conditions since the constraint and intent are captured in task specifications. In this case the reset occurs late in the flyby with limited time between recovery from reset and the next maneuver post flyby. As in the case of the early reset, MEXEC detects the constraint violation for *Instrument Flyby* task when the *Attitude State* is not NADIR post reset. It attempts to execute the same contingency response to restart *GncAttain NADIR* and *Instrument Flyby* task combination but cannot do so while meeting temporal constraints. It therefore attempts additional plan modifiers, in this case it adds a *Instrument Degraded Flyby* task.

MEXEC Planner and Controller

MEXEC consists of separate planning and a control loops to allow plan deliberation time while maintaining responsiveness to currently executing and pending tasks.

The plan update loop consists of a lightweight planner and timeline library that project plans beyond the current control horizon by projecting task impacts and constraints on timelines of shared resources. It updates resources, performs validity checks, and performs plan repair and optimization. If conflicts are detected it applies conflict resolution methods including adding, deleting and moving tasks. The planner commits tasks within the control horizon to the controller, where committing transfers ownership of the activity from the MEXEC planner to the MEXEC controller. The planner can no longer change the activity, however the controller provides updates on state changes of the activity to the planner. This allows the planner to update the activities in the remainder of the plan based on execution. The planner saves the task network at the start and re-loads it if the plan is not improved by the plan update time.

The control loop executes tasks from the current time to the end of the control horizon. Plan constraints and impacts are automatically translated into control conditions. In additional controller only conditions may also be specified for additional monitoring. The controller monitors relevant spacecraft state and processes eligible conditions. It dispatches commands and processes replies. Events received by the controller are queued and processed in the order they are received. In addition it updates the planner with the execution status of each task in real time. When additional tasks are committed, the task network is updated before the next event is processed. The MEXEC control conditions are specified as Boolean expressions. MEXEC allows any combination of AND, OR expression in Boolean expressions with a fixed max number of terms. Most standard Boolean and relational operators are supported in the Boolean expressions. Timeouts and skipping of control conditions are also allowed.

The MEXEC planner uses the timeline library to perform lightweight *validity checks* and compute *valid intervals* to check and repair the remainder of the plan (similar to (Rabideau & Govindjee 1999, Knight R. 2000, Rabideau G.

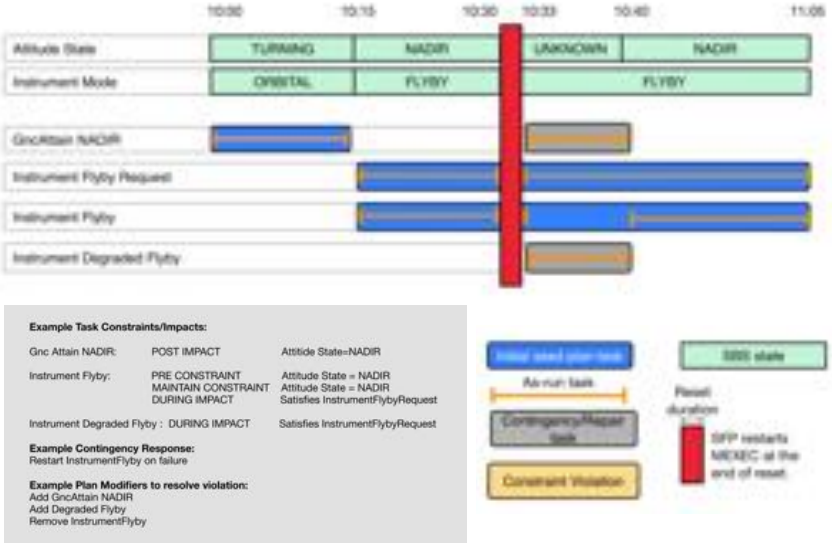

Figure 7: MEXEC response to an early reset.

Figure 8: MEXEC response to an late reset.

2017)). The validity checks return true if the constraint set is satisfied for the given task (or group). Valid intervals are time windows where a task (or group) can be scheduled without violating the constraint set. MEXEC respects the valid intervals of the constraints of the task being scheduled, and the valid intervals of the impacts of the activity being scheduled, with respect to the constraints of other activities. Both impacts and constraints have duration that is considered in the valid interval calculation. Both also have a relative offset from the start and end. The valid intervals for the impact/constraint are translated into valid intervals for the task.

Examples of constraints and impacts are provided in the section on *Constraints and Impacts*. Here we provide examples of validity checks and valid interval calculations performed by the timeline library:

- For an example temporal constraint {*start, end*} of activity *A* must occur {before, after } the {start, end} of activity *B* by *[min, max]*. The validity check is performed by computing the distance between *A* and *B* and comparing to *[min, max]*. The valid interval for a single constraint is computed by adding or subtracting the *min* and *max* distances to the fixed activity to get a range of valid times for the activity being scheduled. For a chain of constraints a shortest path algorithm on the temporal constraint graph may be used.

- For a claim impact (binary semaphore) validity checking is trivial and performed by checking that a new claim does not overlap an existing one. Valid interval calculation is performed by subtracting the intervals that make up the temporal extent from all other claims on the same resource as shown in Figure 9.

- For a state change impact, validity checking is performed by checking that no inconsistent constraint exists between new state changer and the next state changer. Valid intervals are computed by looking between each pair of state changers, and finding the latest inconsistent constraint, include the time interval from the end of the inconsistent constraint to the state changer as shown in Figure 10.

- For a state constraint(s) required over a time range validity checks are performed by checking that the previous state changer (and overlapping changers) change to one of the required states. Valid intervals are computed for each consistent state changer, by including the time range from the consistent changer to the next inconsistent changer as shown in Figure 10.

In the example flyby scenario the *Gnc Attitude* timeline (shown in Figure 11) captures the state impact of various parametrized *GncAttain* tasks such as *GncAttain NADIR*, *GncAttain UVS Scan Init* etc. The *PRE IMPACT* of *GncAttain* tasks is state *TURNING* and the *POST IMPACT* is the attitude commadned by the *GncAttain* task such as *NADIR*. The *InstrumentFlyby* task has a *MAINTAIN CONSTRAINT* that *GncAttitude* must be *NADIR*.

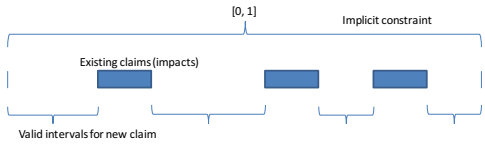

Figure 9: Valid intervals and validity check for claim impacts.

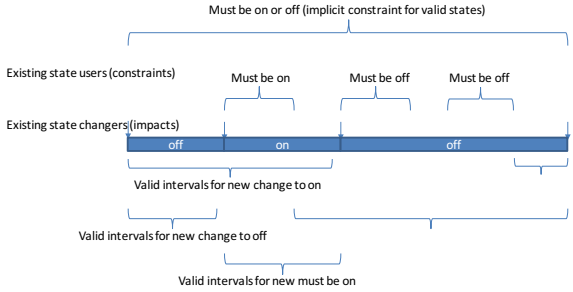

Figure 10: Valid intervals and validity check for state impacts.

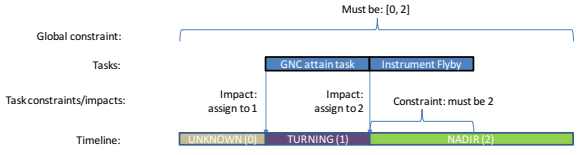

Figure 11: GncAttitude timeline for Instrument Flyby scenario.

Ground Tool

One of the concerns raised by stakeholders was the scope of the task to generate plans with the additional information needed by MEXEC for plan execution. Ground based planning for generating MEXEC can be manual, mixed-initiative or fully automated. A ground tool was developed using the ASPEN planning and scheduling system for the prototype demonstration.

In the manual process a user generates the plan by using the ground tool as a graphical user interface. In the mixed-initiative mode a user generates the plan or fragments of the plan and the planner quickly analyzes the plan for errors or predicted constraint violations. It can suggest a space of resolutions for each. The planner will respect users express demands, even if it does nott understand them. In the fully automatic process, the planner autonomously refines plan until all flight constraints are met. It simultaneously optimizes the plan to include as much science as possible. The autonomous ground planner has different re-planning authority than MEXEC. The ground based planner is typically given much more latitude to rearrange entire plan. The permissions given to MEXEC for in-flight conflict resolution are distinct and typically more limited.

The ASPEN based ground tool allows specification of inputs in various forms:

- Can specify just high level science requests. For example, the input may be for a *Instrument FlybyRquest* at a start time 2028-002T06:00:00 with a duration = 10h. The ASPEN based planner will fill in any required accessory actions such as *GncAttain NADIR*.
- Can combine with explicit low level command invocation. For example, adding an explicit *GncAttain NADIR* at a start time of 2028-002T05:00:00. This is used in case operations have additional information not available to the planner.
- Can specify the initial schedule of activities at a low level. For example, specifying the start time and duration for *GncAttain NADIR*, *Instrument Flyby*, etc. Doing so limits the ability of ASPEN to help with plan generation.
- Can also define tasks from scratch in the editor.

Figure 12 shows the initial user intent to perform a *Instrument FlybyRequest*. The Aspen-based tool automatically detects and resolve the conflicts and proposed the modified plan shown in Figure 14.

Status

Prototype software for MEXEC was developed over four months at a 0.3 full time effort level as part of Europa flight software. MEXEC is designed for multi-mission use with few dependencies on specific flight software and operating system.

In summer 2016 multiple fail-operational scenarios for Europa flyby were demonstrated with MEXEC with resets occurring at different times with GNC maintaining NADIR pointing for flyby science, and GNC slewing for UVS Scan and UVS performing the Scanning. The demonstrations

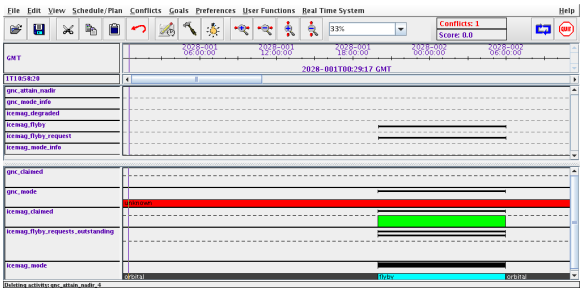

Figure 13: Ground tool with user specified intent.

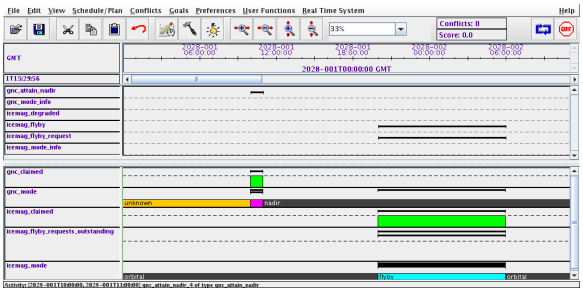

Figure 14: Ground tool proposal for conflict free plan that satisfied intent.

were done in Linux, the Europa flight software simulation workstation test environment (WSTS), and the Europa testbed running MEXEC integrated with Europa flight software on flight avionics hardware. Runs in WSTS and on the Europa testbed were demonstrated with space and time partitioning[1]. The testbed runs were on a 200MHz RAD 750. It took less than 0.1 seconds to resolve plan problems after simulated resets in the testbed for the prototype scenarios.

Since Europa flight software is in early development GNC, Thermal and Instrument software was simulated. However, the actual flight software interfaces were used to communicate between MEXEC and the simulated modules and the Europa flight software state buffer interface was used for monitoring spacecraft state.

A ground tool was also developed to generate MEXEC plans and MEXEC was demonstrated end-to-end executing a plan generated by the ground tool.

Conclusions and Lessons Learned

The MEXEC prototype demonstrated that lightweight planning and execution can provide fail-operational capability in a flight-like environment. It provides the capability for checking constraints on-board for the executing and remaining plan. Unlike a large proportion of the alternate flight approaches it does not require special cases for flight software resets occurring at different times. In addition it provides plan enhancement options. MEXEC can add, move, and remove tasks if given permission.The prototype also demonstrated capability with time and space partitioning on Europa

[1]Each application software runs in a partition with its own memory space and dedicated time slot

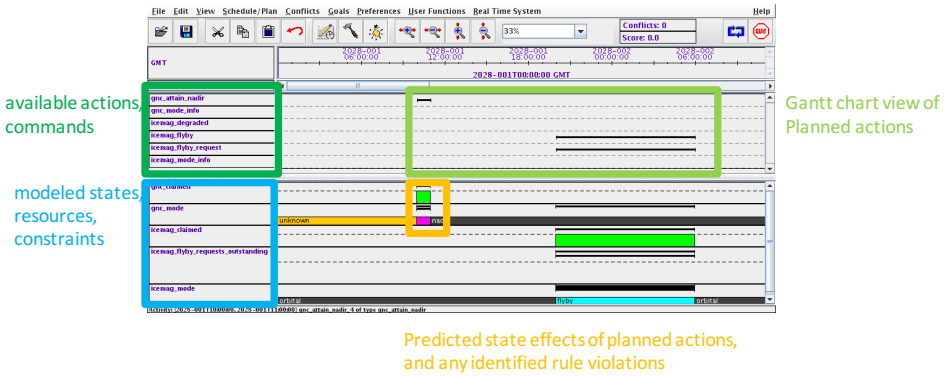

Figure 12: ASPEN based ground tool interface.

system testbed. It also provided a preliminary feasibility assessment for flight and ground interfaces and tools. MEXEC is currently integrated into Europa Flight Software, but is not in the baseline. The timeline library from MEXEC has also been ported to the Mars 2020 rover project for use in the Mars 2020 Onboard Planner (OBP) (Rabideau G. 2017).

A common concern was the ability to perform verification and validation of flight planning and execution software. Another key concern for operators was the need to develop user friendly interfaces to specify and manage MEXEC tasks.

There is a tremendous amount of prior work in the planning and execution literature (Gat 1998, R. Simmons 1998, Muscettola 1998, S. Chien 1999, S. Chien 2000, Rabideau & Govindjee 1999, J. Frank 2001, A. Jonsson 2000, V. Verma 2005, R. Rasmussen 2005, **?**, V. Verma 2006, T. Estlin 2007, **?**). Strides have been made in on-board autonomy in Earth observing missions, however deep space missions have continued to use sequencing based approaches as the primary means for spacecraft control. Due to radiation hardening requirements deep space missions have very limited computing (the Europa mission is baselined to use a RAD750 processor), and preference for human oversight. Unlike a number of the previous approaches the charter for the MEXEC prototype was not a new automated planning driven architecture or event driven scripting language, but instead a new component in low risk heritage flight software and operations approach. MEXEC code was written completely from scratch. It inherits from the literature and was designed to meet current flight software requirements and constraints, and allow human operators the ability to control the level of automated plan update at a high level of granularity.

Acknowledgments

We would like to thank Julia Bell, Jeff Biesiadecki, Joe Carsten, Seung Cheung, Steve Chien, John Day, Dero Gharibian, Corey Harmon, Bill Heventhal, John Ibanez, Mitch Ingham, John Kwok, Mary Lam, Mark Maimone, Lloyd Manglapus, Peter Meakin, Shannon Mihaly, David Mohr, Bob Rasmussen, Trina Ray, Kathy Schimmels, Joel Signorelli, Kim Steadman, Tim Weise, and Katie Weiss for the feedback they provided during the development of this work.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Copyright 2017 California Institute of Technology. U.S. Government sponsorship acknowledged.

References

[A. Jonsson 2000] A. Jonsson, P. Morris, N. M. K. R. B. S. 2000. Planning in interplanetary space: Theory and practice. In *In Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Aspen, CO, April 2000*.

[Gat 1998] Gat, E. 1998. On three-layer architectures. In *In D. Kortenkamp, R. Bonnasso, and R. Murphy, editors, Artificial Intelligence and Mobile Robots, Boston, MA, 1998. MIT Press*.

[J. Frank 2001] J. Frank, A. Jnsson, R. M. D. S. 2001. Planning and scheduling for fleets of earth observing satellites. In *Venue: In Proceedings of the Sixth International Symposium on Artificial Intelligence, Robotics, Automation and Space*.

[Knight R. 2000] Knight R., R. G. C. S. 2000. Computing valid intervals for collections of activities with shared states and resources. In *AIPS*.

[Muscettola 1998] Muscettola, N. 1998. Remote agent: To boldly go where no ai system has gone before. In *Artificial Intelligence, 103(1-2):5?48*.

[R. Rasmussen 2005] R. Rasmussen, M. Ingham, D. D. 2005. Achieving control and interoperability through unified model-based engineering and software engineering. In *AIAA Infotech@Aerospace Conference. Arlington, VA*.

[R. Simmons 1998] R. Simmons, D. A. 1998. A task description language for robot control. In *In IEEE/RSJ Intelligent Robotics and Systems Conference, Vancouver Canada*.

[Rabideau & Govindjee 1999] Rabideau, G., K. R. C. S. F. A., and Govindjee, A. 1999. Iterative repair planning

for spacecraft operations using the aspen system. In *In Artificial Intelligence, Robotics and Automation in Space*.

[Rabideau G. 2017] Rabideau G., B. E. 2017. Prototyping an onboard scheduler for the mars 2020 rover. In *International Workshop on Planning and Scheduling for Space (under review)*.

[S. Chien 1999] S. Chien, R. Knight, R. S. G. R. 1999. Integrated planning and execution for autonomous spacecraft. In *IEEE Aerospace Conference, Aspen CO, March 1999*.

[S. Chien 2000] S. Chien, R. Knight, A. S. R. S. G. R. 2000. Using iterative repair to improve responsiveness of planning and scheduling. In *InProceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Aspen, C0*.

[T. Estlin 2007] T. Estlin, D. Gaines, C. C. R. C. B. B. M. J. I. N. R. A. 2007. Increased mars rover autonomy using ai planning, scheduling and execution. In *IEEE International Conference on Robotics and Automation (ICRA 2007). Rome, Italy*.

[V. Verma 2005] V. Verma, A. Jnsson, R. S. T. E. R. L. 2005. Survey of command execution systems for nasa spacecraft and robotics. In *Plan Execution: A Reality Check? Workshop at The International Conference on Automated Planning & Scheduling (ICAPS)*.

[V. Verma 2006] V. Verma, A. Jnsson, C. P. M. I. 2006. Universal executive and plexil: Engine and language for robust spacecraft control and operations. In *American Institute of Aeronautics and Astronautics Space 2006 Conference*.