"In OBP We Trust": Verification and Validation of the M2020 On Board Planner Flight Software

Shreya Parjan
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91011
shreya.parjan@jpl.nasa.gov

Dan Gaines
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91011
daniel.m.gaines@jpl.nasa.gov

***Abstract*—The Mars 2020 On Board Planner (OBP) entered primary operations on the Perseverance rover on October 5, 2023. OBP addresses a productivity challenge faced by every Mars rover mission thus far: predicting resource usage in the dynamic Martian environment. Historically, operators have leveraged conservative models to generate constrained schedules for when activities must execute onboard. On Board Planner enables Perseverance to respond to variations in activity execution and expected state, ultimately reducing conservatism in planning and addressing significant deviations from the initial schedule during plan execution. Effectively infusing OBP's capabilities and reaping its benefits is predicated on operators trusting that the flight system is and will remain safe in the dynamic exploration environment with OBP in control. This paper describes the Verification and Validation (V&V) campaign performed to provide that assurance, with a focus on flight software testing for OBP's first release. After an overview of On Board Planner itself, we place the V&V campaign in the context of earlier efforts to V&V autonomy for space applications, describe the behaviors tested and the tools and workflows to do so, and hone in on two case studies that demonstrate the OBP V&V process in action. We conclude by examining results, challenges, and key takeaways from the campaign and how they may inform future efforts to provide assurance for onboard autonomy.**

Table of Contents

1. Introduction

The use of the Mars 2020 On Board Planner (OBP) flight software (FSW) component on the Perseverance rover marks a paradigm shift in Mars rover operations. OBP replaces Master Sub-Master (M/SM) mode, which has been used for tactical planning in all Mars rover missions thus far. In M/SM operations, a *master sequence* contains the commands to perform the activities planned for a given operating sol (Figure 1). The master sequence manages fixed rover wake-ups and shutdowns and activates other sequences necessary for activity execution. It may span multiple Martian days (*sols*).

979-8-3503-0462-6/24/$31.00 ©2024 IEEE

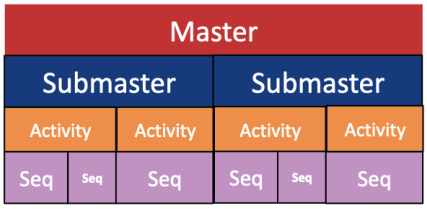

Figure 1. Composition of a master sequence, which contains commands to perform activities in Master Sub-Master mode. The master sequence manages sequences necessary for activity execution (sub-masters), which are responsible for lower-level activities and their constituent sequences.

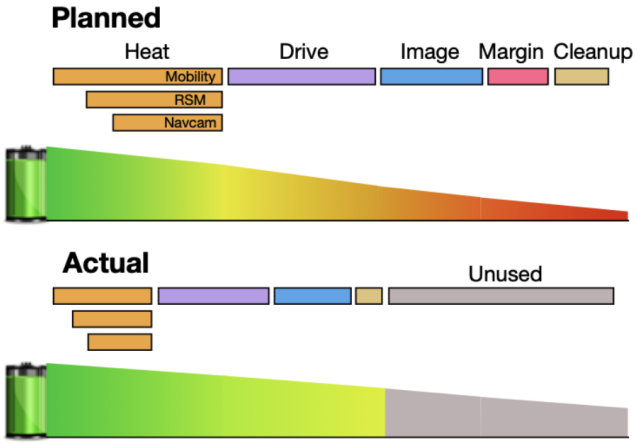

Figure 2. Example of how M/SM planning conservatively models resources for activity execution (a drive, in this case). In execution, excess time and energy are under-utilized [1].

The sequences activated by the master are referred to as *sub-masters*. Sub-masters may activate additional sequences to support the execution of planned activities.

By way of sub-masters, the master sequence provides the rover with fixed, step-by-step instructions on when and how to execute activities. In M/SM, the timing and ordering of activities, rover sleep periods ("naps"), and heating are fixed. Further, M/SM does not support constraints related to activity timing, inter-activity dependencies, or activity parallelism. To prevent over-depletion of the battery and under-heating of devices (e.g., motors and electronics), M/SM also leverages highly conservative models to predict the duration and onboard resources used by planned activities. A case

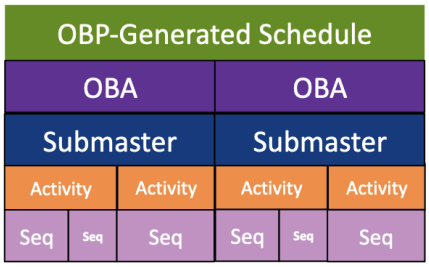

Figure 3. Composition of an On Board Plan File. Note that in contrast to Master Sub-Master mode, On Board Activities (OBAs) serve as the base unit considered by OBP flight software during scheduling and execution.

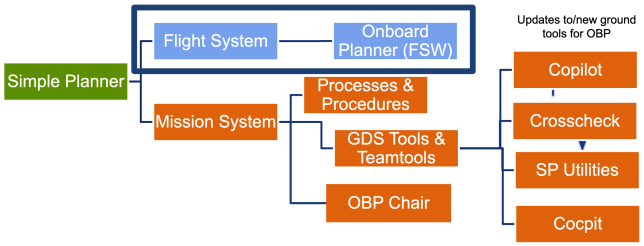

Figure 4. OBP consists of the flight system component of Simple Planner (boxed). Simple Planner encompasses OBP and the mission system updates, tools, and workflows necessary to plan with OBP instead of M/SM.

study of Mars Science Laboratory (MSL) Curiosity rover campaigns revealed that rover energy and time resources were often under-subscribed in M/SM planning [2]. As a result, activities were under-run by 28% on average, representing a significant impact on rover productivity when comparing projected and actual resource drain. The M/SM planning approach is prohibitively conservative and leaves little room to respond to deviations from predicted execution, such as when an activity ends early and does not require its allocated margin (Figure 2).

On Board Planner enables the Perseverance rover to respond to variations in activity execution and deviations from expected state (i.e., warmer temperatures or higher state of charge). An *OBP-Generated Plan File* (OBPF) is uplinked to the vehicle to replace the traditional master sequence. The OBPF captures operator specifications of desired, flexible activity execution windows as well as both inter-activity and resource constraints. Sequences called by the master sequence in M/SM are replaced by *On Board Activities* (OBAs) (Figure 3). OBAs are the base unit utilized by the On Board Planner flight software when scheduling and updating the initial schedule in response to deviations (i.e., environmental, resource, or activity-related) during execution of the OBPF. The On Board Planner flight software autonomously manages wake-ups, shutdowns, and heating based on resource models and activity execution states, rather than on fixed timing in the plan.

OBP fits under the broader umbrella of Simple Planner (Figure 4), which also includes the ground tools and workflows operators need to develop OBPFs and interface with the flight software [3]. With a two-phase incremental roll-out, Simple Planner will transform the planning and rover operations process to allow for higher-level specification of desired activities in place of historical M/SM operations [4]. Operational use of On Board Planner as part of Simple Planner's first release (SP1) began on October 5, 2023. In its first release, OBP reduces energy consumption and saves energy for a higher baseline state of charge in later sols. The second release (SP2), with an expected roll-out of Spring 2024, will allow OBP to use previously saved energy in the current sol to execute optional or expanding activities.

OBP supports flexible execution based on ground-specified constraints and leverages event-based replanning to respond to significant schedule deviations during execution [5]. The flight software consists of two tasks: (1) *plan*, which generates plans given operator input and current vehicle resources using a greedy, non-backtracking scheduler, and (2) *plan controller* (*planc*), which supports flexible execution by altering the start time of activities in response to schedule deviations. *planc* pulls activities earlier if resources allow and predecessors complete early or pushes them later if a predecessor runs long or resources are currently insufficient (Figure 5).

On Board Planner will need to make scheduling decisions without the immediate oversight of human operators on behalf of a sophisticated vehicle in the dynamic Martian environment. Thus, its success is inextricable from the trust that rover operators have in it—trust which is earned through thorough flight software verification and validation. For our purposes, *verification* asks whether the developed flight software satisfies its specification. *Validation* assesses whether the FSW meets its requirements and motivating OBP goals. In this paper, we hone in on the role of the flight software testing program in ensuring that the flight system is and will remain safe with OBP in control for SP1. Section 2 describes previous work on V&V campaigns for autonomous flight software. Section 3 provides an overview of OBP's flight software. Section 4 delves into the requirements, workflow, and tools leveraged in the testing campaign. Section 5 presents and evaluates examples of how we exercised the V&V process. Finally, Section 6 describes results from the testing campaign, the impact of OBP on Perseverance operations thus far, and key takeaways from the V&V conducted in support of the SP1 release.

2. Related Work

An extensive body of previous work documents the importance of, challenges associated with, and criteria for the verification and validation of autonomous software for space applications [6–10]. For one, communication opportunities are limited by frequency and bandwidth, so an agent must respond to variations in activity execution or resources without human operators readily in the loop. Flagship class missions such as M2020 are also costly and complex. Associated spacecraft such as the Perseverance rover include sophisticated science instruments that must be operated within complex operating constraints. Finally, onboard computing power is typically heavily constrained—Perseverance's Rover Compute Element (RCE) is powered by a BAE RAD750 that runs at 133 MHz and provides 266 MIPs that OBP must share with other FSW tasks of varying criticality [1].

Previous work verifying and validating autonomous flight software for use in space applications has leveraged both formal and informal methods, along with requirements-based testing. For the EO-1 and IPEX spacecraft, developers

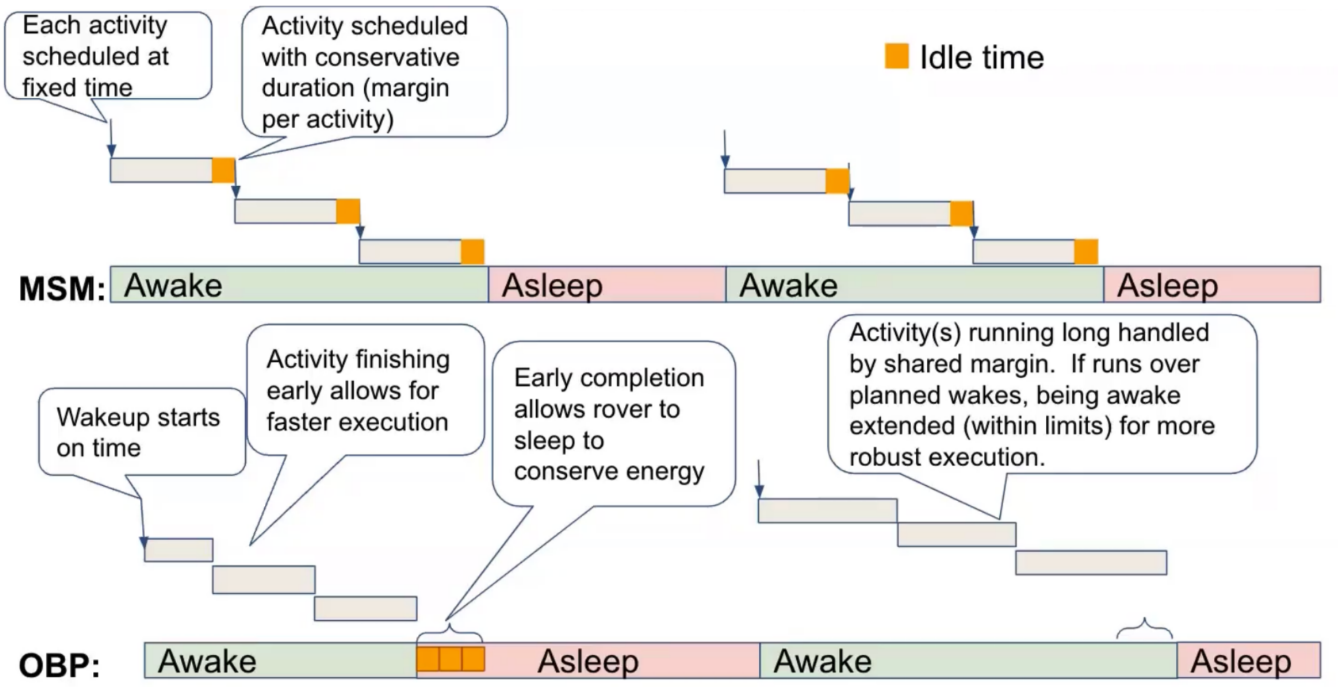

Figure 5. Comparison between Master Sub-Master and OBP execution. With OBP in control, when activities complete early, the rover can go to sleep earlier to conserve energy. Additionally, if an activity runs long, the rover can stay awake longer to accommodate them if resources allow.

validated the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) software with tests that utilized baseline, extrema, and stochastic parameter values, in addition to an environmental testset that captured variation points in both execution and stochastic parameter values [11, 12].

M2020 flight software developers (including members of the OBP FSW development team) leveraged formal methods previously used in flight software V&V for the Curiosity rover. These included model checking and static analysis [13]. Informal methods that informed OBP algorithm design decisions analyzed techniques for flexible execution, switch groups, and energy scheduling [5, 14, 15]. In Section 3, we provide an overview of the flight software architecture that resulted from these implementation decisions.

The campaign to test the flight software in anticipated operations-like scenarios, stress cases, and off-nominal scenarios is the focus of our discussion in the rest of this paper. Testing provided traceable documentation of how the OBP FSW assures flight system safety while meeting its design specification. Key challenges specific to the OBP testing campaign included accounting for behavioral uncertainty introduced by the flexible execution paradigm, constraining the unbounded problem space of conditions and scenarios OBP would encounter in flight to a finite number of testable units, and ensuring appropriate interaction with external, non-OBP FSW modules (particularly to manage heating [16] and honor operational constraints).

3. OBP Flight Software

Under SP1, On Board Planner helps the Perseverance rover recover idle time, increase robustness to variations in activity execution, and reduce conservatism when modeling onboard resource consumption. OBP's constituent modules *plan* and *planc* (Figure 6), along with the *timeline* library, interface with Perseverance's other subsystems to manage onboard resources (i.e., power, heating, and data volume). Collectively, they assure the safety of the vehicle, accommodate communication windows, and enforce instrument operations constraints. Here, we provide a high-level overview of the flight software to contextualize our discussion of the V&V campaign. More details and analysis specific to FSW development can be found in [1, 17, 18].

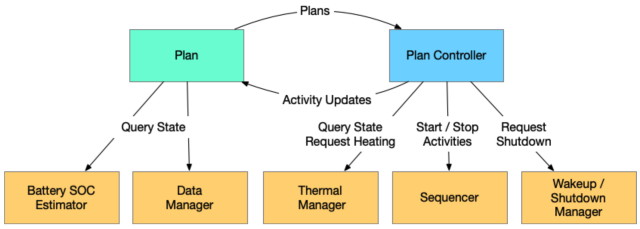

Figure 6. Interactions between the *plan* and *planc* tasks of OBP flight software and other flight software components. The initial plan is uplinked by operators and updated by *plan*based on feedback from Plan Controller (*planc*) [1].

The *plan* task schedules activities based on vehicle resource models and operator input. Figure 7 outlines *plan*'s scheduling algorithm. The greedy algorithm performs no lookahead or backtracking when placing activities in priority order. Instead, once it finds a spot that honors resource and inter-activity constraints, it places each activity and moves onto the next. Such simplifications reduce the ability of the planner to find an optimal schedule but significantly reduce computation

cost, thereby allowing more CPU availability for other flight software tasks. To model resource consumption, particularly for battery state of charge (SOC), OBP leverages *Cumulative Rate Timelines*. Represented with the *timeline* library, a Cumulative Rate Timeline contains a set of impacts modeling the rate of change (i.e., watt-hours) or value (i.e., watts) when the rover goes to sleep and wakes back up. When a new activity is considered, valid intervals to schedule it are generated based on these resource timelines [15].

planc supports flexible plan execution and event-based replanning by altering the start time of activities in the schedule generated by *plan*. If the rover is idle, *planc* may pull a future activity earlier if it is eligible to start earlier. If an activity is ineligible to start at its scheduled start time, *planc* may delay (push) the activity. For example, if *planc* finds actual temperature readings for thermal zones used by an activity are colder than predict, the preheat for the heated activity may be extended to allow it to complete. If the activity is delayed too long from its original scheduled start time, it will be vetoed and re-scheduled by *plan* as execution time constraints allow. For activities that need heating before and during use, OBP discretizes thermal intervals to reduce the search space of all possible times when heating may start. In these discretized, 10 minute intervals, temperatures are assumed constant.

The flight software also leverages *event-based rescheduling*, in which rescheduling is triggered by a significant deviation from the original schedule (i.e., activities ending significantly early/late, getting vetoed, being aborted while in progress based on some termination criteria, or completing with failure). Because schedules take time to generate, an activity scheduled to start at the current time may be past its start time once scheduling completes. Instead, *plan* uses a sliding buffer-like *commit window* during which new activities cannot be scheduled to start. When activities are committed during the window, *planc* takes over and decides whether the activity is ready to start based on eligiblity criteria like interactivity dependencies and necessary resources. These design considerations increase OBP's responsiveness to variations in onboard and environmental conditions and ultimately support more effective utilization of vehicle resources.

4. TESTING CAMPAIGN

The majority of the software testing for OBP's first release was performed in Spring 2022, well after Perseverance's landing in February 2021. As a result, the testing campaign was informed by over a year of Perseverance surface operations on Mars under the Master Sub-Master paradigm. We leveraged realistic scenarios informed by hundreds of sols of planning and could replicate operating conditions in simulations to generate realistic telemetry and other data products. Figure 8 provides an overview of the workflow for the testing campaign:

1. The release's scope was translated into FSW goals and requirements that could be targeted and verified by sets of related test cases, called *Verification Activities* (VAs). VAs helped break OBP functionality into testable components. To that end, we maintained a matrix listing each requirement and the VAs that provided test evidence for it. Individual requirements were often targeted by a combination of VAs, ensuring comprehensive coverage across test cases from different test sets. During *test reviews*, the V&V team and FSW engineer(s) discussed proposed tests for a VA and whether they provided adequate coverage of the associated requirements.
2. Each VA was executed through its test cases. Implementing the tests involved generating OBPFs and tailoring execution (i.e., injecting temperature values for relevant thermal zones, modifying parameters, etc.) as necessary to trigger behaviors of interest.
3. In addition, we used a lightweight OBP simulation tool called *plansim* to perform a preliminary run of the scheduling algorithm and basic FSW execution to refine test cases before the more time-intensive, higher-fidelity software simulation.
4. Once all tests were scripted, the V&V engineer performed a *Run for Record* (R4R) to execute all tests and generate their associated data products (warnings, *Event Records* (EVRs), and other simulated telemetry). Failure detection and explanation require that test engineers go through the results captured in these data products and look for off-nominal or unexplained behavior that deviates from expectation.
5. The as-run procedure, data products and any relevant notes on FSW behavior or anomalies were compiled into an *Activity Report* (AR) presented at a *data review* for approval from the V&V leads. Once approved, the testing was counted towards closure evidence for its associated requirements in the matrix of VAs and verification items.

Our workflow relied on the following resources:

- *DOORS Next Generation (DNG)*: IBM's DNG supported requirements management and the organization and editing of requirements, VAs, and test case artifacts. DNG allowed V&V engineers to update test case and VA statuses and link tests to the requirements for which they provided closure evidence.
- *Workstation Testset (WSTS)*: WSTS is a high-fidelity, software-only virtual testbed that allows for multiple users to execute simultaneous testing on the M2020 flight software in the absence of actual flight hardware. WSTS runs six times faster than real-time and allows the user to pause/resume/step through its software simulation which includes modeling resource consumption, dispatching commands, and parsing simulated telemetry.
- *Jupyter Notebook*: A templatized Jupyter notebook standardized the set-up process for our virtual testbeds and the configuration of the OBP flight software for test development and execution. V&V engineers implemented test cases for each VA in a unique notebook, resulting in a centralized, executable script when conducting runs for record. We utilized Jupyter notebooks to document test cases and perform runs for record to make implementation decisions clear and results easily viewable to future reviewers. Jupyter notebooks are also used to document software testing across the M2020 project, so utilizing them helped ensure consistency with already-completed testing.
- *Mission System Testbed (MSTB)*: Occasionally, additional validation was necessary beyond what WSTS could provide. As a higher fidelity testbed equipped with two Rover Compute Elements (RCEs) but no actuators, the MSTB served this purpose. For instance, performance tests and tests requiring higher fidelity thermal interfaces (particularly for hardware that is active when the rover is asleep) took place on the MSTB.
- *Vehicle System Testbed (VSTB)*: The VSTB rover is a nearly identical, Earth-based version of the Perseverance rover. It was used by the Simple Planner team for end-to-end tests during early systems and operations integration [4].

Beyond the nominal SP1 V&V campaign, V&V engineers were occasionally called upon to provide test evidence confirming that component-level fixes to OBP flight software

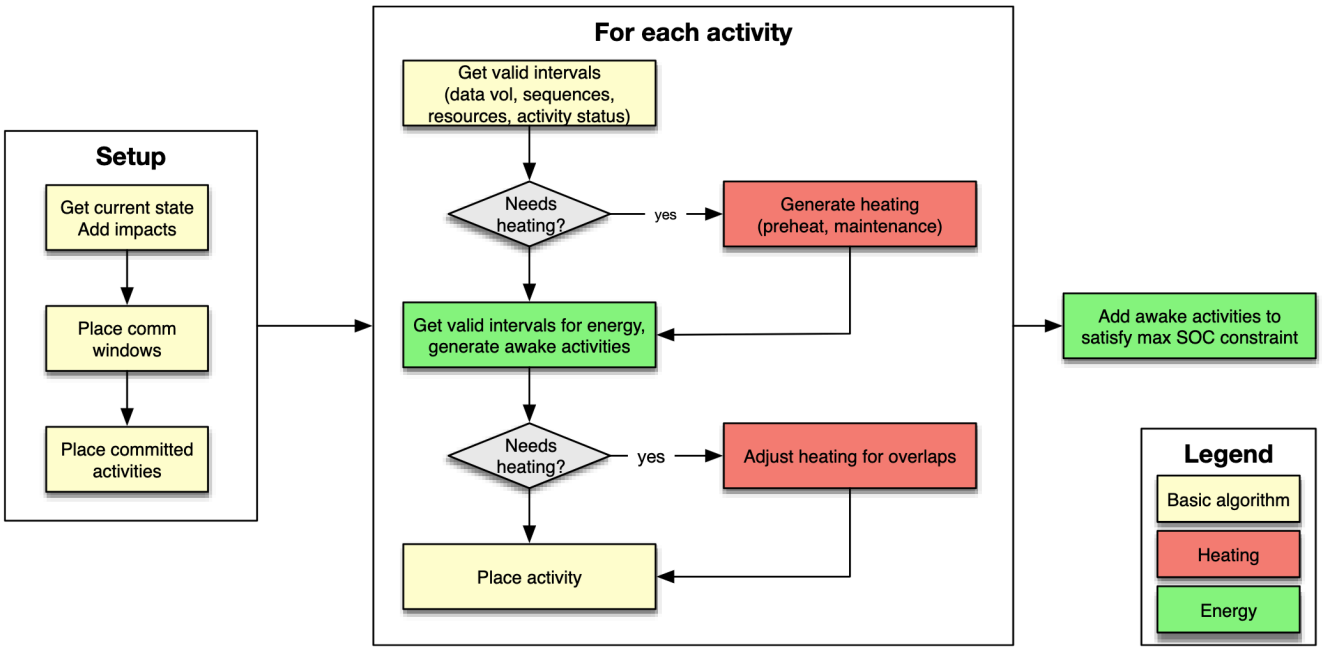

Figure 7. OBP scheduling algorithm [17].

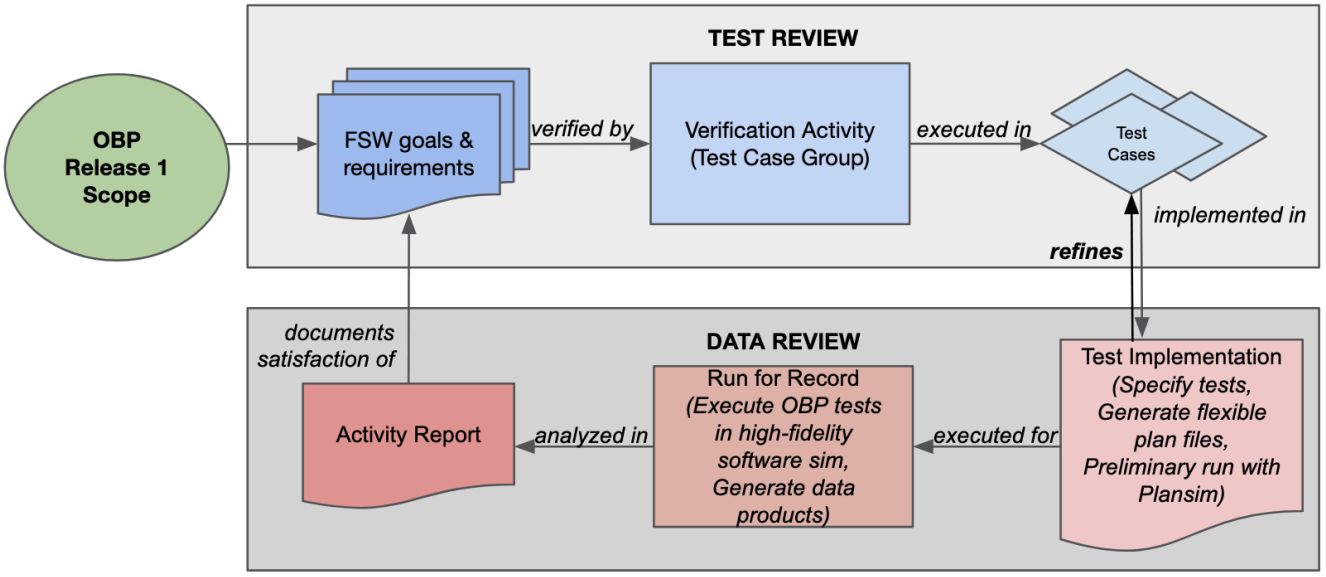

Figure 8. Workflow for SP1 testing campaign. Key differences from traditional V&V include the ability to iterate on proposed test cases based on results from the *plansim* lightweight simulation tool and executing OBP flight software in WSTS, a high-fidelity software sim.

adequately responded to bugs caught during earlier testing. Anomaly descriptions and closure evidence were documented as part of the mission's Problem Failure (anomaly) Reporting (PFR) process, with analysis and intermediate work documented in associated JIRA tickets. Multiple key PFRs closed through V&V addressed OBP management of onboard heating, a particularly tricky functionality due to the need to monitor temperatures prior to starting heating, assorted thermal tables used by OBP to calculate target temperatures and preheat durations, and uncertainties around heating spanning rover sleep cycles [16]. Further, during the SP1 campaign, V&V engineers performed regression testing on the S8 release of flight software, which included updates to incorporate validated PFR fixes to OBP components into the release. The V&V team's regression approach involved rerunning a large subset of procedures on the new release in WSTS and documenting any anomalies or deviations from previous runs that used S7 flight software. Long-term, the team is interested in narrowing down a suite of representative procedures to run automated regression testing on and alleviate the manual workload.

5. EXAMPLE VERIFICATION ACTIVITIES

Over the testing campaign for SP1, 154 requirements were satisfied across 32 VAs. Among many other behaviors, these Verification Activities included tests of the following capabilities:

1. *Resource Management*: Resources may include *vehicle resources* (time, power, energy), *activity resources* (resource claims to prevent unsupported parallelism, sequence engines, communications window compatibility, and *operator-provided constraints* (handover batter state of charge, minimum or maximum battery state of charge, peak power limits).
2. *Inter-Activity Dependencies*: Activities could require others in the plan to have not started, completed with success, or either finished (i.e., completed with success or failure) or not been scheduled. A sample OBP behavior covered by this capability is ensuring that post-drive imaging does indeed occur *after* a drive.
3. *Heating*: This covers behaviors like managing preheat activities to get zones to operational temperature for use in heated activities, ensuring maintenance heating runs while a heated activity is executing, merging heating for neighboring heated activities in a plan that use the same thermal zone, and handling heating that occurs during periods where the rover is inactive and asleep.
4. *Plan Transforms*: Since testing took place well after Perseverance began operating on Mars, we were able to tailor actual plans executed on the vehicle to execute with OBP in control.

To offer insight into the testing campaign, we discuss a subset of the VAs for the SP1 release of OBP flight software in greater detail.

Example 1: Execution Status and Dependencies

Context— Since OBP enables flexible execution, it is important for operators to be able to constrain when activities can occur relative to one another to prevent plans from deviating drastically from initial operator specifications. The SP1 release achieves this by allowing activities to have dependencies on the execution status of other activities. In particular, activities may be specified (in Conjunctive Normal Form [CNF]) to require others to have finished, completed with success, either finished or not scheduled, or a combination of these. The following requirements are associated with the VA:

1. The Flight System shall provide telemetry indicating the completion status of activities.
2. The Flight System shall be capable of autonomously executing activities based on pre-defined criteria.
3. Among other criteria, activity eligibility to execute depends on the CNF on the activity's activity dependency constraints being satisfied.
4. When evaluating a CNF task dependency term, if the referenced activity does not exist in the current plan nor in the last two active plans, the FSW shall evaluate the term to FALSE, regardless of whether the term is negated.
5. The flight software shall track (among others for SP2) the following execution statuses for each activity: FINISHED, COMPLETED WITH SUCCESS, FINISHED or NOT SCHEDULED.
6. Dependencies may be expressed to the FSW with a bitmask. However, ground tools should only present them at a high level to avoid requiring operators to expend effort constructing an appropriate grouping.

Test Cases—The test cases focused primarily on comparing behavior when activities had FINISHED dependencies vs. FINISHED or NOT SCHEDULED dependencies on others in the plan. In addition, we explored actual ops-like scenarios. For instance, to schedule mobility activities with OBP, operators use compound dependencies that include dependencies on distinct activities such that one must be COMPLETED WITH SUCCESS and the other must be FINISHED. The full list of test cases developed to target the requirements for the use of execution status and dependencies in SP1 is as follows:

1. Given a chain of activities A-D with either FINISHED or "FINISHED or NOT SCHEDULED" dependencies between A and B, B and C, and C and D (i.e., B requires A to have finished, C requires B to have finished, and so on), let C run past its attributed duration and into its cutoff time, ultimately ending up aborted (i.e., it is marked as FINISHED and its cleanup sequence will run). We expect that in either case, activities A, B, and D will still complete with success.
2. Given a chain of activities A-D with either FINISHED or "FINISHED or NOT SCHEDULED" dependencies between A and B, B and C, and C and D, let the plan execute through the completion of activity D. All dependencies will be satisfied and all activities will complete with success.
3. Given a chain of activities A-D with either FINISHED or "FINISHED or NOT SCHEDULED" dependencies between A and B, along with C and D, force A to complete with failure by sending a specific command and let C fail to schedule because its corresponding sequence has not been uplinked. We expect that A completes with failure and C fails to schedule, meaning that B and D execute to completion in the FINISHED or NOT SCHEDULED case, but D fails to schedule in the FINISHED case.
4. Given a chain of activities A-D with either FINISHED or "FINISHED or NOT SCHEDULED" dependencies between A and B, B and C, and C and D, let B map to a sequence that runs longer than its attributed duration, leaving insufficient time for C to schedule before its cutoff time while still remaining within D's execution window. Ultimately, A, B, and D schedule to completion while C fails to schedule in the FINISHED or NOT SCHEDULED case. In the FINISHED case, A and B schedule to completion while C and D fail to schedule.
5. Given three OBAs: A, B, and C, let C require A to have completed with success *and* B to have finished. Demonstrate both success and failure cases based on whether A completes with success and/or B finishes.
6. Given a plan file with two OBAs, A and B, let B require A to have completed with success or some activity that doesn't exist in the plan, X, to have not have scheduled. Ensure activity A's duration is short enough that activity B gets committed before activity A ends. Send a command to for A to complete with failure during execution but after B has been committed. Because X does not exist, the dependency constraint that B has on it will automatically evaluate to false, even though X is indeed not scheduled. We also repeated the same test but with B only requiring X to not have scheduled.
7. Given a chain of activities A-D with COMPLETED WITH SUCCESS dependencies between A and B as well as C and D. Let C also have a FINISHED or NOT SCHEDULED dependency on B and let A complete with failure through commanding. Ultimately, A's completion with failure will cause B to fail to schedule. C and D will still schedule and execute to successful completion.

Execution and Results—To give a sense for what test case execution looks like based on schedules generated in the Jupyter notebook run, we present results from the failure case for test 5 described above (Figure 9).

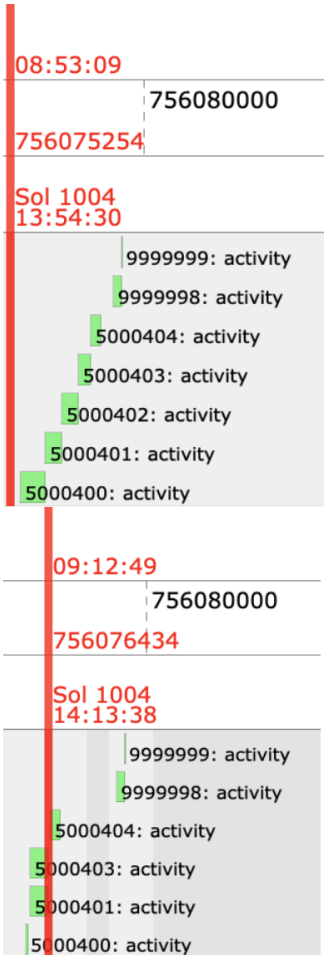

Figure 9. First and fourth schedules generated during test case 5 (failure case) execution.

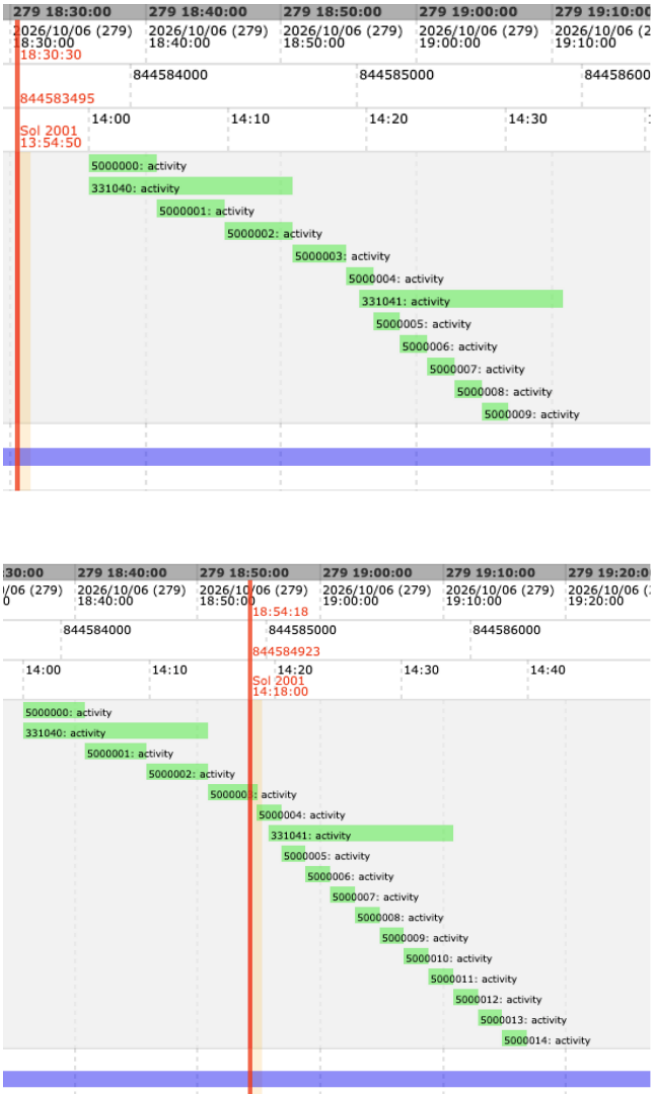

Figure 10. First and second schedules generated during test case 1 execution. The two comm windows not counted to the considered set have IDs 331040 (highest priority) and 331041 (lowest priority). All other activities are operator-specified and numbered in priority order.

In Figure 9, activities A, B, and C correspond to the blocks labeled 5000400, 5000401, and 5000402 respectively. In the initial schedule, all three are present. Activities A and B have no dependencies specified, while activity C requires A to have completed with success and B to have finished. Additionally, activity D (ID 5000403) requires C to have finished or not been scheduled at all and activity E (ID 5000404) similarly requires D to have status FINISHED or NOT SCHEDULED.

Between the schedules at the times depicted, activity A completes with failure (due to commanding manually sent by the test engineer). At that time, activity C drops from the schedule (see the lower portion of Figure 9) because the portion of its compound dependency that requires A to have completed with success is unsatisfied. This test also demonstrates that even though activity C is not scheduled, activities D and E can still execute. Through the use of the FINISHED or NOT SCHEDULED dependency, OBP can prevent activities from jumping around significantly in the plan (for instance, D and E could not be scheduled before C). Additionally, the structure will prevent a single point of failure (in this case, C dropping from the schedule) from taking out subsequent activities that depend on it. In practice, this specific dependency structure was tested to represent the way the mobility subsystem would structure their activities with OBP in control.

Example 2: Considered Set

Context—The idea of the ”considered set,” a subset of the yet-to-be executed activities from an activity hopper of 100 activities specified by operators, has long been of interest to OBP flight software developers [19]. The capability was initially pushed to SP2, but system-level integration tests identified the need for a larger number of operator-specified activities for multi-sol plans than previously expected. As a result, the considered set capability was re-prioritized for SP1 roll-out and V&V was required to test OBP’s ability to satisfy the following requirements:

1. The FSW shall maintain a hopper of operator-provided activities, moving activities from the hopper to the considered set when room is available in the considered set. The size of the hopper is defined as the maximum number of operator-provided activities (100) minus the size of the considered set.
2. The FSW shall store a subset of operator-provided activities in a considered set, the set of activities considered when

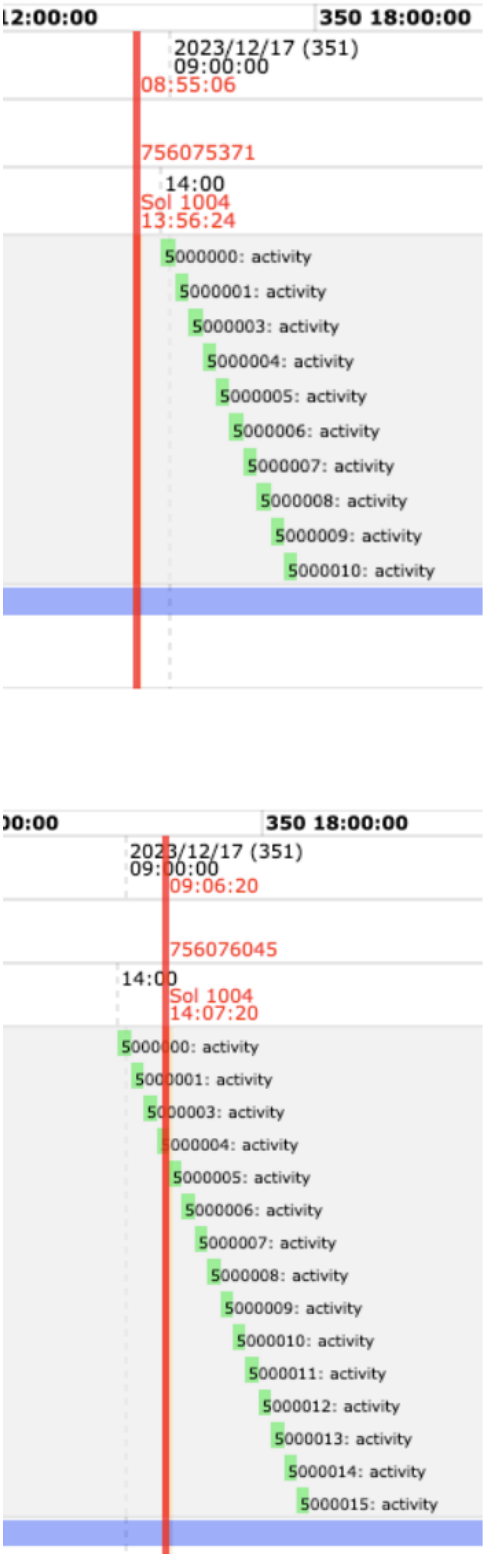

Figure 11. First and second schedules generated during test case 4 execution. All activities were numbered based on their priority, with activity 5000002 marked unschedulable at the time of plan activation.

generating a schedule.
3. The FSW shall remove an activity from the considered set after it has completed execution.

Test Cases—Each of the following test cases were evaluated on a plan file containing 20 activities. The size of the considered set was lowered by updating the *max_considered* parameter. The first test also included a subcase where two communications windows were included as high and low priority activities in the plan file to demonstrate that non-operator provided activities (such as communications windows) were not included in the considered set.

1. Considered set of 10 activities.
2. Considered set of 5 activities.
3. Activity that ultimately fails to schedule counts toward considered set.
4. Unschedulable activities (i.e., activities the planner will not/no longer consider for scheduling) do not count toward considered set.

Overall, we expected that 1) each subsequent schedule would grow by the number of activities in the considered set that were committed and executed and 2) subsequently scheduled activities would continue to reflect activity scheduling priorities.

Execution and Results—Here, we provide a subset of the test execution results for cases 1 (with comm windows) and 4. Execution completed as expected for test case 1 (Figure 10). In particular, the initial scheduling cycle placed the first 10 MS-specified OBAs and the comm windows. The second scheduling cycle was triggered when OBA 5000004 was committed. It added the next five highest-priority priority non-comm OBAs to the plan. In the following scheduling cycle, the last five OBAs in the plan file were scheduled. As evidenced by the number of OBAs added to the plan between scheduling cycles, comm windows were not counted toward the *max_considered* count.

In test case 4 (Figure 11), OBA 5000002 was marked unschedulable because its execution time range spanned a time that had elapsed by the time the plan was activated. *max_considered* was 10 for this test. To confirm that execution completed as expected, note that in the first schedule, the eleventh OBA by priority (5000010) was scheduled (Figure 12). Given a considered set size of 10, if activity 5000002 counted towards *max_considered*, then activity 5000010 would not have appeared in the initial schedule. Figure 12 also illustrates that once the fifth mandatory OBA was committed, the second scheduling cycle placed the next five activities by priority order (5000011-5000015).

The examples presented here are only a small subset of the complete SP1 V&V effort and evidence how testing validated the flight software's ability to support complex scheduling and execution requirements. In addition to testing spanning 32 SP1-specific Verification Activities and 154 associated requirements, the campaign also iterated on adaptations of actual sol 100-600 plans run on Perseverance. In particular, plans from the Rapid Traverse campaign [10] that spanned sols 385 to 410 were modeled with OBP in control to derive differences in drive distance between OBP and M/SM. Using the associated telemetry to inform simulations, V&V engineers were able to analyze future energy savings possible under the Simple Planner paradigm.

	SP1	SP2
Scheduling Freedom	• OBAs stay in same rover wake cycle and order is preserved. • Time changes only based on previous OBAs running shorter or longer than predict.	• OBAs can move more within the plan as long as they honor their specified constraints.
Key Benefit	• Reduces idle awake time.	• Reduces idle awake time. • Uses power savings for additional activities.
Constraint-based Planning Support	• Pinned (fixed start time) activities • Execution windows • Dependencies	• Pins • Multiple execution windows • More sophisticated dependencies • Preferred time
New Capabilities	• Similar to M/SM	• Allows drives to expand to save power • Optional activities added if resources allow

Figure 12. Comparison of SP1 and SP2 capabilities.

Furthermore, 57 anomalies were identified and resolved, with testing in the loop throughout the process. After a V&V engineer documented anomalous behavior during testing, the anomaly was either dispositioned as meriting a flight software fix or simply a workaround such as compliance with a new flight rule for tactical operations. Due to the complexity of interfacing with the thermal subsystem, multiple critical anomalies pertained to OBP management of preheat and maintenance heating [16]. If a flight software update was required and made by FSW engineers, the test engineer would then re-test the scenario that triggered the originally reported anomaly and report on whether the fix resolved it.

6. Discussion and Future Work

On Board Planner has successfully been in control on the Perseverance rover from October 5, 2023 to the time of writing, with the exception of a return to Master Sub-Master operations during solar conjunction between November 8, 2023 and December 1, 2023. The flexible execution paradigm motivated a precedent shift in rover operations. Some of the key returns during this period include:

1. Significant state of charge energy gifts per sol due to OBP-enabled savings (higher gains expected in fall/winter). However, currently 20% of energy savings are shunted due to state of charge limits. In SP2, we will be able to leverage energy savings to perform more science in the current sol.
2. Longer drives, including the drive to the solar conjunction parking spot for the rover which ran 70m longer than predict and freed up 1 sol for additional activities.
3. More precise heating leading to improved science quality.

The testing campaign played a crucial role in discretizing the problem space through the workflow presented in this paper. Testing also ensured that not only can OBP reap the rewards that motivated its development, but that it can be relied on to ensure vehicle safety and compliance with operational constraints. With the SP2 V&V campaign well under-way, we are building on the expansive swath of behaviors tested during SP1 V&V to support more sophisticated behaviors such as using energy savings to execute additional activities and modeling more complex activity constraints (Figure 12).

7. Conclusion

Equipping the Perseverance rover with autonomous scheduling and execution capabilities is one of several precedent-shifting achievements for onboard autonomy demonstrated during the Mars 2020 mission [20]. The robust verification and validation campaign for OBP's first release was critical to the success of Simple Planner thus far. We posit that OBP is well-equipped to reduce the need for extremely conservative resource modeling, support more flexible activity execution while remaining mindful of operational constraints, and recover time and energy on the Perseverance rover as part of SP1, the first phase of Simple Planner operations. The testing campaign discussed in this paper details how we exercised the flight software under realistic day-to-day and off-nominal scenarios alike while leveraging sophisticated, flight-like simulations. Ultimately, we hope that the V&V done in support of On Board Planner's commissioning for use on Perseverance can serve as a fruitful example for future efforts to commission onboard autonomy for use in flight.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors would also like to recognize the rest of the On Board Planner SP1 V&V team for their contributions: Stephen Kuhn, Andrew Plave, Gregg Rabideau, Kevin Reich, Ansel Rothstein-Dowden, Daniel Tran, Nicholas Waldram, and Sean Wenzel.

REFERENCES

[1] D. Gaines, G. Rabideau, V. Wong, S. Kuhn, E. Fosse, and S. Chien, "The mars 2020 on-board planner: Balancing performance and computational constraints," in *Flight Software Workshop*, February 2022. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/presentations/OBP-FSW22-2022-01-07.pdf

[2] D. Gaines, G. Doran, H. Justice, G. Rabideau, S. Schaffer, V. Verma, K. Wagstaff, V. Vasavada, W. Huffman, R. Anderson, R. Mackey, and T. Estlin, "Productivity challenges for Mars rover operations: A case study of Mars Science Laboratory operations." *Technical Report D-97908, Jet Propulsion Laboratory*, January 2016. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/gaines-report-roverProductivity.pdf

[3] A. Connell and M. Hurst, "Ground software to support autonomous onboard scheduling for mars perseverance rover," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–10.

[4] R. Siegfriedt, S. Chien, D. Gaines, S. Kuhn, J. Hazelrig, J. Biehl, A. Connell, R. Francis, and N. Waldram, "Mars 2020 onboard planner - update and preparations for operations," in *17th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2023)*, October 2023. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/M2020-SP-ASTRA-2023.pdf

[5] J. Agrawal, W. Chi, S. Chien, G. Rabideau, D. Gaines, and S. Kuhn, "Analyzing the effectiveness of rescheduling and flexible execution methods to address uncertainty in execution duration for a planetary rover," *Robotics and Autonomous Systems*, vol. 140, p. 103758, 2021. [Online]. Available: https://doi.org/10.1016/j.robot.2021.103758

[6] C. Pecheur, "Verification and validation of autonomy software at NASA," National Aeronautics and Space Administration, Tech. Rep., 2001.

[7] R. Cardoso, G. Kourtis, L. Dennis, C. Dixon, M. Farrell, M. Fisher, and M. Webster, "A review of verification and validation for space autonomous systems," *Current Robotics Reports*, vol. 2, 09 2021.

[8] M. S. Feather, "Hallmarks of an autonomous space system's development and V&V," in *2022 IEEE International Conference on Assured Autonomy (ICAA)*, 2022, pp. 129–136.

[9] S. Chien, "Formal methods for trusted space autonomy, boon or bane?" in *NASA Formal Methods Symposium*, May 2022. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/chien-nfm-2022.pdf

[10] M. S. Feather and A. Pinto, "Assurance for autonomy - JPL's past research, lessons learned, and future directions," in *IEEE International Conference on Assured Autonomy, ICAA 2023, Laurel, MD, USA, June 6-8, 2023*. IEEE, 2023, pp. 97–105. [Online]. Available: https://doi.org/10.1109/ICAA58325.2023.00022

[11] B. Cichy, S. Chien, S. Schaffer, D. Tran, G. Rabideau, and R. Sherwood, "Validating the autonomous EO-1 science agent," in *International Workshop on Planning and Scheduling for Space (IWPSS 2004)*, Darmstadt, Germany, June 2004. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/cichy-iwpss2004.pdf

[12] S. Chien, J. Doubleday, D. Tran, D. Thompson, K. Wagstaff, J. Bellardo, C. Francis, E. Baumgarten, A. Williams, E. Yee, D. Fluitt, E. Stanton, and J. Piug-Suari, "Flight validation of HyspIRI IPM concepts on the intelligent payload experiment (IPEX) cubesat," in *HyspIRI Product Symposium*, Greenbelt, MD, June 2014. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/chien-hyspiri2014-flight.pdf

[13] G. J. Holzmann, "Mars code," *Commun. ACM*, vol. 57, no. 2, p. 64–73, feb 2014. [Online]. Available: https://doi.org/10.1145/2560217.2560218

[14] J. Agrawal, W. Chi, S. A. Chien, G. Rabideau, S. Kuhn, D. Gaines, T. Vaquero, and S. Bhaskaran, "Enabling limited resource-bounded disjunction in scheduling," *Journal of Aerospace Information Systems*, vol. 18:6, pp. 322–332, 2021. [Online]. Available: https://doi.org/10.2514/1.I010908

[15] W. Chi, S. Chien, and J. Agrawal, "Scheduling with complex consumptive resources for a planetary rover," in *International Conference on Automated Planning and Scheduling (ICAPS 2020)*, Nancy, France, October 2020. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/chi-icaps2020-wakesleep.pdf

[16] S. Parjan and D. Gaines, "Towards trusted Mars autonomy: V&V of the M2020 on board planner's thermal management capabilities," in *Proceedings of the International Workshop on Planning and Scheduling for Space (IWPSS)*, July 2023. [Online]. Available: https://drive.google.com/file/d/1-KMJ4OYcpYv47jdGh8nK5dmXDm1hE9Wj/view

[17] D. Gaines, S. Chien, G. Rabideau, S. Kuhn, V. Wong, A. Yelamanchili, S. Towey, J. Agrawal, W. Chi, A. Connell, E. Davis, and C. Lohr, "Onboard planning for the Mars 2020 Perseverance rover," in *16th Symposium on Advanced Space Technologies in Robotics and Automation*, June 2022. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/M2020-OBP-ASTRA-2022-Final.pdf

[18] S. Kuhn, "From determinism to emergence: Systems engineering a step change in execution on Mars," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–10.

[19] G. Rabideau, V. Wong, D. Gaines, J. Agrawal, S. Chien, S. Kuhn, E. Fosse, and J. Biehl, "Onboard automated scheduling for the Mars 2020 rover," in *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation for Space*, ser. i-SAIRAS'2020. Noordwijk,

NL: European Space Agency, 2020. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/M2020-OBP-i-SAIRAS2020-camera.pdf

[20] V. Verma, M. W. Maimone, D. M. Gaines, R. Francis, T. A. Estlin, S. R. Kuhn, G. R. Rabideau, S. A. Chien, M. M. McHenry, E. J. Graser, A. L. Rankin, and E. R. Thiel, "Autonomous robotics is driving perseverance rover's progress on mars," *Science Robotics*, vol. 8, no. 80, p. eadi3099, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adi3099

BIOGRAPHY

Shreya Parjan *received a B.A. in Mathematics and Computer Science from Wellesley College in 2021. Shreya currently serves as a Flight Software Verification & Validation Engineer for the Mars 2020 On Board Planner and an On Board Planner Chair supporting the roll-out of Simple Planner in Mars 2020 tactical operations.*

Dan Gaines *received a Ph.D. in Computer Science from the University of Illinois at Urbana Champaign in 1998. He has been the Deputy Cognizant Engineer for M2020 Flight Software and a developer for MSL and Europa Clipper flight software. He has over 15 years of mission operations experience including MER, MSL and M2020.*