Risk-Reduction Autonomy Implementation to Enable NASA Artemis Missions

Fernando Figueroa
NASA Stennis Space Center
Autonomous Systems Laboratory
Stennis Space Center, MS 39529
fernando.figueroa@nasa.gov

Lauren Underwood
NASA Stennis Space Center
Autonomous Systems Laboratory
Stennis Space Center, MS 39529
lauren.w.underwood@nasa.gov

Jonathan Morris
D2K Technologies
5062 Nighthawk Way
Oceanside, CA 92056
jon.morris@d2ktech.com

Mark Walker
D2K Technologies
5062 Nighthawk Way
Oceanside, CA 92056
mark.walker@d2ktech.com

Rane Brown
Lockheed Martin Space
12257 S Wadsworth Blvd
Littleton, CO 80125
rane.brown@lmco.com

***Abstract*—To achieve NASA's Artemis program mission objectives a high level of autonomy that is ubiquitous throughout the systems that are being developed will be necessary. The autonomous systems of Artemis will require a distributed autonomy capability, with autonomous systems organized functionally in a hierarchical architecture, where systems at higher levels of the hierarchy have authority over systems at lower levels. The challenge of developing autonomy technologies and Concepts of Operations (ConOps) for Artemis has been undertaken by the NASA Gateway Working Group. This group has developed requirements, architectures, ConOps, and interface control documents (ICDs), in the context of a hierarchical distributed architecture that includes the following: a Vehicle System Manager (VSM) that autonomously manages the entire Gateway; Module System Managers (MSMs) that autonomously manage each module; and System Managers (SMs) that autonomously manage systems within a module (i.e., ECLSS). A substantially high level of autonomy needs to be achieved by each element of the hierarchy (VSM, MSM, SM) to meet requirements for uncrewed operations; this includes conditions that will have minimal and/or delayed ground intervention (i.e., requirements for sustainability for months of operation without crew or ground support). To advance an implementation of this autonomy design (Gateway Autonomy Design — GAD), a collaboration was established between the Autonomous Systems Laboratory (ASL) at NASA Stennis Space Center and Lockheed Martin. The objectives of this partnership were the following: (1) to implement autonomy at the VSM, MSM, and SM levels; (2) to implement communications among a VSM, 2 MSMs, ORION (a visiting vehicle somewhat equivalent to a module) and 1 SM (a power system), and (3) test autonomous operations with representative use cases. A SM backed by a high-fidelity simulation was created to facilitate demonstrations of use cases that originated in a system of a module. Communication between the VSM and MSMs was implemented according to Concepts of Operations and Interface Control Documents (ICDs). Demonstrations were conducted to address nominal and off-nominal operations and multi-module interactions with the VSM. Additionally, user interfaces were created to provide awareness about ongoing processes and results while enhancing the demonstration. Demonstrations included the following use cases: (1) Orion as visiting vehicle registers with VSM; (2) VSM reschedules a module's timelines when another module's MSM task fails; and (3) a module's Power System Manager (PSM) demonstration that included component failure diagnostics, tracing component failure to effected components, which in turn, reports failure information up to the VSM for acknowledgement and display. This paper will describe the detailed technology and autonomous systems developed, and the integrated multi-module demonstrations conducted. Also, challenges that must be met to fully implement the GAD defined by Gateway will be addressed.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. AUTONOMY DESIGN: HIERARCHICAL DISTRIBUTED ARCHITECTURE 2
3. TOPOLOGY OF AUTONOMOUS MODELS DEVELOPED 3
4. CONCEPT OF OPERATIONS 3
5. TASK AND TIMELINE DEVELOPMENT AND EXECUTION 5
6. INTEGRATION ACROSS MODULES USING SBN .. 6
7. DEMONSTRATION 7
8. SUMMARY AND CONCLUSIONS 10
ACKNOWLEDGMENTS 11
REFERENCES 11
BIOGRAPHY 12

1. INTRODUCTION

Autonomous systems are becoming more common place as self-driving vehicles are now on the streets and roads, and unmanned air vehicles (UAVs) are in plain sight over streets, events, and in military applications. These systems carry out some activities autonomously. A self-driving vehicle does go from a start location to an end location on its own, without intervention from a human. But is the system truly autonomous? One definition adopted at NASA in 2015 states "Autonomy is the ability of a system to achieve goals while operating independently of external control" [1]. However, these goals generally address very specific activities and do not incorporate overall autonomous behavior of a system or system of systems. This means that systems do not behave autonomously with respect to self-management. For example, a self-driving vehicle considers a wide range of scenarios that may require changing a planned route due to external factors, like road closings. However, these decisions generally do not include comprehensive information such as whether a part or a sensor has failed and the vehicle no longer has capability to continue or to avoid risks. These system-level autonomy behavior factors take on a much bigger role as systems become more complex (for example, multiple systems with operational/behavior dependencies), when time delays for operator intervention become too long (e.g., operations on Mars, or even on the Moon), and/or when risks of system failures or anomalies are high and have catastrophic consequences.

For NASA"s Artemis missions, autonomy has been perceived as an entire system exhibiting autonomous behavior, and not as independent or isolated capabilities that carries out only individual autonomous activities. Additionally, when designing system(s) to perform autonomously for specific activities, there should be a foundation for implementing overall autonomous behavior. In this way, any autonomous activity is then able to use the resources from a foundational autonomous behavior of the entire system, resulting in the ability to implement higher level autonomy functionality, that can evolve systematically and not be added on as an afterthought.

Another key consideration in executing autonomy is how it is accomplished. In order to exhibit autonomous behavior, a system must be able to analyze, reason, make decisions, as well as schedule and execute tasks on its own. The classic approach to achieve this autonomous capability is accomplished by having experts working off-line who consider every possible scenario a specific system will encounter — under both nominal and off-nominal operations. This approach generally assumes that nominal operations will achieve the desired goals, and that all cases of off-nominal conditions are recognized. It also relies on defining and providing corresponding mitigations that are necessary to continue the mission, to modify the mission, and/or to abort the mission. This approach is implemented using a few technologies that help keep track of the cases being considered and aid in reasoning about the cases. Unfortunately, this technique is very difficult to apply, and is extremely costly, even for systems that are considered only moderately complex. For example, consider a power system for a space habitat module where this approach would necessitate the ability to detect and consider the consequences of failure of every switch, every wire, every bus, every sensor, and every battery cell. Moreover, this solution inevitably misses critical and/or potentially life-threatening cases and/or circumstances. This approach is referred to as Brute-Force Autonomy (BFA).

The Autonomous Systems Laboratory (ASL) at NASA Stennis Space Center has been developing the technology, processes, and expertise to implement autonomy by transferring the knowledge and methodologies for analysis, and decision-making of experts to the system itself. This implementation has been coined as "Thinking" Autonomy (TA). The technology is embodied in the NASA Platform for Autonomous Systems (NPAS) [2]. With NPAS, there are four primary pillars required to enable TA:

1. A comprehensive description of the system. This could be considered an augmented description beyond what is enabled by SysML. For example, this might include fidelity to the schematic level, generic physics and other models, concept definitions and self-identification, inter-dependencies, and topologies [3].
2. A comprehensive description of the mission to be carried out.
3. Integrated System Health Management (ISHM) strategies, including anomaly detection, diagnostics, prognostics, and comprehensive and timely state awareness [3].
4. Autonomy strategies (for example, determination and use of redundancy and repeat actions).

This paper describes accomplishments made as part of an avionics risk reduction activity conducted in a collaboration effort between NASA Stennis Space Center and Lockheed Martin. This activity culminated in a demonstration of autonomy implemented in the context of the NASA Gateway [4], [5] distributed hierarchical autonomy [6] design and the corresponding avionics hardware/software architecture (network protocols and message definitions). Prior to this work, a prototyping activity was demonstrated in 2019, in collaboration with Northrop Grumman, to demonstrate autonomous operations for an individual habitat module [7]. That work provides background about the degree of system-level autonomy that is required for Artemis systems and operations, and contrasts the difference between that degree of autonomy to systems that can only discretely perform some activities autonomously. This paper describes results of a project that extends autonomous operations beyond an individual space habitat, and not only includes multiple modules, but also demonstrates and increase in the fidelity of the systems' models and the complexity of mission operations with distributed autonomous entities, as well as incorporating ground operations and crew.

In this paper the following topics will be covered: section 2, Autonomy design hierarchical distributed architecture, describes hierarchical distributed architecture encompassing an overall autonomous manager (NASA Vehicle System Manager (VSM)), Module System Managers (MSMs), and System Managers (SMs), as well as managers for ORION consistent with NASA Gateway autonomy design; section 3, Topology of autonomous modules developed, includes details on the NASA Stennis and Lockheed Martin modules as well as ORION; section 4, Concepts of Operations, includes description of the Concept of Operations (ConOps) addressing the functions at the top 3 levels of the autonomy hierarchy; in section 5, Task and timeline development and execution, includes NPAS and LM's capabilities and implementations; section 6, Integration across modules using the Software Bus Network (SBN); section 7, Demonstration, includes an overview of use cases presented; and section 8, Summary and Conclusions, provides a high level representation of accomplishments.

2. AUTONOMY DESIGN: HIERARCHICAL DISTRIBUTED ARCHITECTURE

Figure 1 represents a topology for hierarchical distributed autonomy, and corresponding systems that encompass a notional autonomous space habitat.

The elements that constitute members of a hierarchical distributed architecture are autonomous themselves [8]. The level each occupies in the hierarchy defines their level of authority. At the top is the VSM with authority over all the elements below. At the next level are the modules, and below each module are the systems in the modules. Each autonomous entity has an individual mission, and all participate in achieving the overall mission managed by the VSM. The overall mission of the VSM entails the following objectives:

1. Resource management (e.g., power, telemetry, communications)
2. GNC
3. Health/fault management
4. Mission plan, timeline, and execution management

Although the project only addressed objectives 3 and 4, the infrastructures in NPAS and LM cFS applications support all objectives.

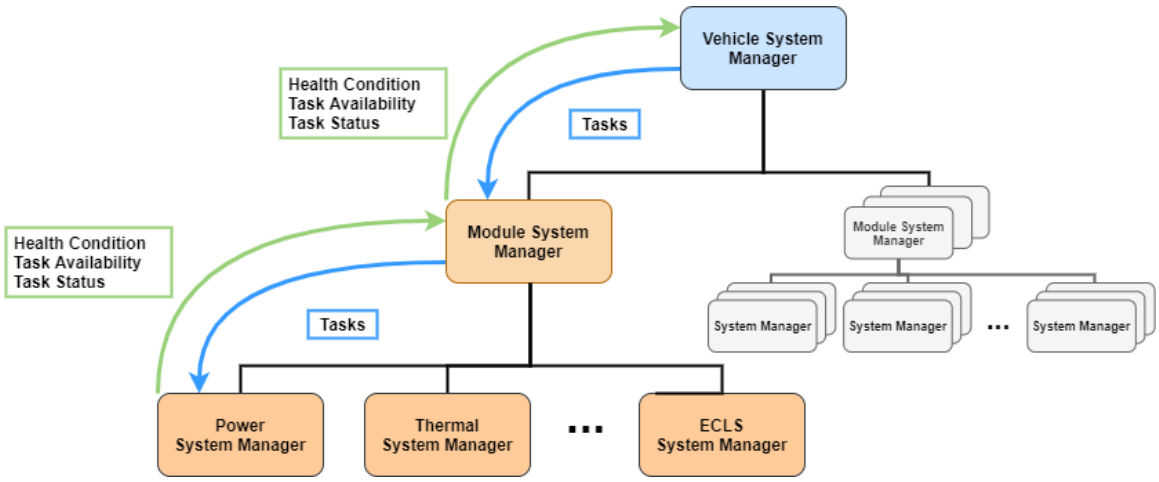

Figure 1. Topology for hierarchical distributed architecture

3. Topology of Autonomous Models Developed

For this collaborative effort, NASA Stennis, which includes the D2K support contractor team, was responsible for the development of the overall Gateway Vehicle System Manager (VSM), a Module System Manager (MSM) for a Logistics module, and a Power System Manager (PSM) using NPAS to rapidly build and deploy autonomous solutions consistent with the NASA Gateway's autonomy concept of operations.

The VSM is the autonomous agent responsible for the overall management of Gateway's operations to achieve its specified mission objectives. To accomplish this, the VSM is responsible for:

- Planning and scheduling timelines of goals to be achieved and tasks to be executed by the Module System Managers.
- Coordinating Module-to-Module and Gateway-to-ground communications and operations
- Responding to failures within the Gateway and adapting as-needed to achieve its mission.
- Serving as the primary interface between the Gateway and crew and ground operations.

For this demonstration the VSM focused on the following key objectives: (1) MSM-to-VSM registration for permanent Gateway modules; (2) docking and registration for visiting vehicles; (3) executing pre-defined timelines of Tasks scheduled for multiple MSMs; (4) responding to failures in lower level autonomous system managers; and (5) creating crew interfaces for monitoring Gateway's operations.

The Logistics MSM comprised of three NPAS applications — the MSM, PSM, and ESM — reusing core Gateway autonomy framework provided by NPAS.

- Contained Environmental Control and Life Support System (ECLSS) System Manager
- PSM
  - PSM had a detailed system schematic driven by SOCRRATES
  - Detailed failure management system

Lockheed Martin was responsible for the development of the Habitat Module, Orion as a Visiting Vehicle and the Software Only Crew Exploration Vehicle Risk Analysis Test Environment Simulation (SOCRRATES) simulation environment. Existing environments and tools were leveraged with new development and updates made to support the goals of the demonstration.

The Habitat Module used the NASA developed open source flight software architecture called cFS (Core Flight System). cFS provides a reusable framework that allows for mission specific application development without redesigning the entire flight software architecture for each new program [12]. For the autonomous VSM demonstration, Lockheed Martin developed three new cFS applications: (1) MSM, (2) Science, (3) Environmental Control and Life Support System (ECLSS). The most important of these was the MSM which was responsible for communication with the VSM to register capabilities and resources, as well as receive tasking. After receiving instructions from the VSM, the MSM would schedule tasks for the Science and ECLSS applications. For example, the ECLSS application would be tasked with maintaining and reporting air quality. The Science application would be tasked with conducting a recurring experiment on a specific schedule and reporting results.

For this demonstration, Orion as a Visiting Vehicle primarily leveraged existing simulation environment SOCRRATES/OrionSim (Orion Simulation) which was developed for the Orion program (see section 7 for additional detail). Additionally, new capabilities developed for this demonstration consisted of creating a controlling MSM that could communicate with the Orion flight software. The focus for this use case was a docking scenario which required Orion as the Visiting Vehicle communicating with the VSM once the vehicle was authorized to proceed with docking, and then sharing related docking telemetry. After successful docking, the Orion MSM would conduct standard registration with the VSM. See section 6 for additional details on the registration process.

4. Concept of Operations

The Concepts of Operations (ConOps) defines how these objectives or functions are carried out as coordinated plans and timelines across all elements of the hierarchy. The

following are high level tenants for operation:

1. Modules are registered with the VSM, providing information about their resources and tasks.
2. Modules keep the VSM updated about their health, faults, diagnostics, and tasks.
3. The VSM defines the overall mission with tasks and timelines to be carried out by all the members of the overall spacecraft.
4. Tasks address every function to carry out the overall nominal mission and individual member missions.
5. Tasks address every function to carry out any off-nominal operation to resolve anomalies and continue mission operations.

The project and demonstration included use cases that address each of the above operation tenants involving the modules and systems that encompass the reference representative of a Lunar Gateway.

Incorporation of Digital Twin Domain Representations

One of the features that the NPAS-based autonomous operations system managers utilize for task management as well as for detection and resolution of anomalies is the use of a software domain reference model — also known as a "digital twin". NPAS supports the creation of such domain specific models using built-in and extensible component model libraries. As discussed in Section 1, these model libraries are typically designed to match the level of representation included in provided schematics and Interface Control Documents (ICDs).

The resulting detailed models provide many benefits to the autonomous system software. One benefit is the provision of a dynamic system state model that the autonomous software can reference in real-time. This enables the software to make use of both topological and operational context when employing its reasoning logic. This is a two-fold benefit since the detailed domain representation actually facilitates the delivery of the TA described in Section 1. The implementation of TA is made easier and maintainable through the use of model object attributes, class methods, and the instantiation of relationships and associations between the model components (e.g., the ability to reason about components that are currently "upstream", or "feeding-power-to" other components).

NPAS Digital Twin domain representations are ontological (e.g., they are composed of model constructs that map well to the ontologies present within the systems, processes, and equipment), topological (e.g., they include the components, the interconnections, and the associations between the components in an operational context), and dynamic (e.g., they are populated with live telemetry data and real-time state assessment regarding health and operational availability). The representations are incorporated into the autonomous system manager software architecture seamlessly so that regardless of domain, all applicable measurement data can be properly integrated into a state model.

Other aspects pertinent to the use of digital twins for autonomous operations include the design of user interfaces and model visualizations. Even though the digital twin domain representation is a graphical model that can be useful for visualizing state, generally the NPAS-based autonomous systems are built with proxy displays that allow for the transmission and conveyance of important health states and parameters directly to users. The same is especially true within distributed, hierarchical systems of systems where communication interfaces are incorporated to allow for the transmission of state information between system managers. In this way, the VSM UI can readily display summary information derived from the PSM digital twin.

Example: Implementation of the PSM Digital Twin

The first step in creating the PSM digital twin was to gain access to the target LM spacecraft power system schematics and ICDs. From these drawings, the NPAS mission developer can begin constructing the detailed domain models — following basic modeling guidelines for ensuring that the resulting models are complete, fully connected, and fully configured. Figure 2 shows a schematic of one of the power system module components, and Figure 3 shows the corresponding PSM digital twin.

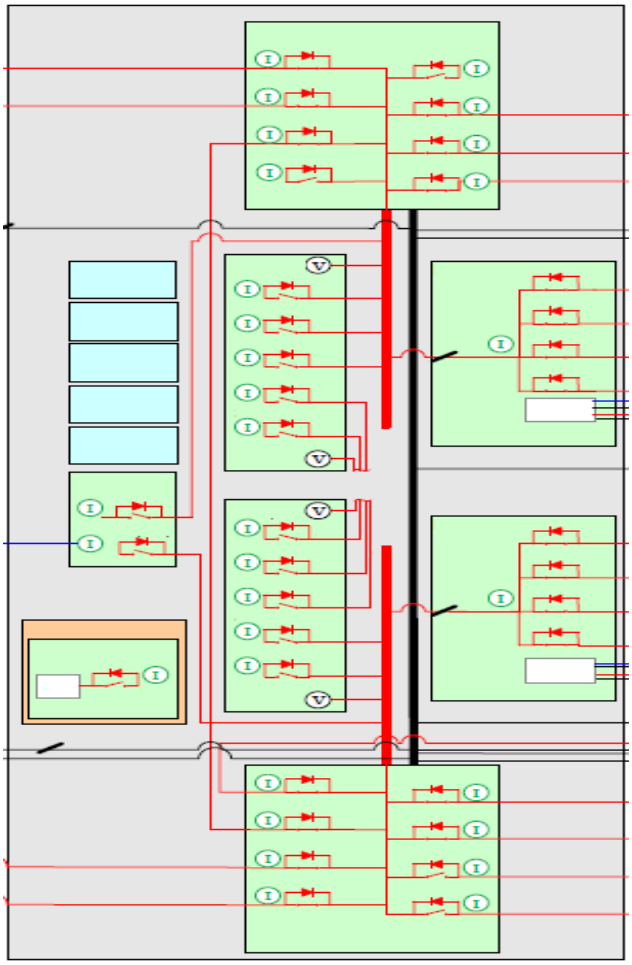

Figure 2. LM Power System PDU Schematic Example

The PSM digital twin domain model was constructed using the NPAS electrical equipment domain modeling libraries. These libraries provide reusable access to component classes associated with electrical generation and distribution systems (both AC and DC). The NPAS equipment libraries are extensible, which allows NPAS mission developers to create new classes and specialized subclasses as required. A screenshot of the use of NPAS electrical modeling palettes during modeling of another PSM PDU is shown in Figure 4.

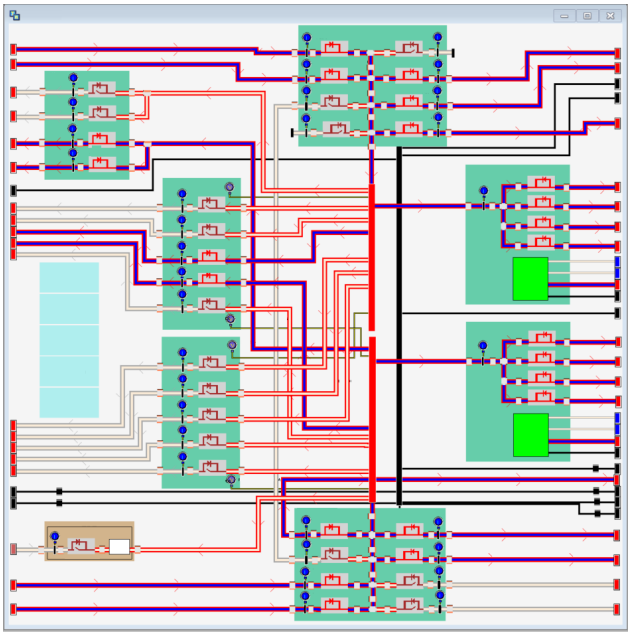

Figure 3. PSM PDU Digital Twin

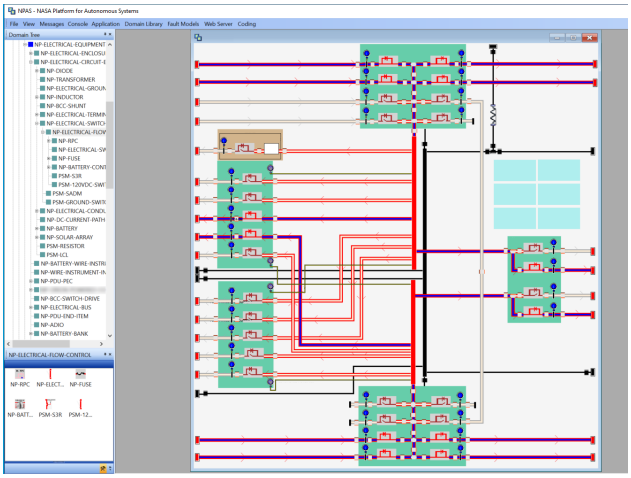

Figure 4. Ontological Modeling of Power Systems

The second step to modeling and deploying the PSM digital twin is in the incorporation and application of generic failure modes and effects (Failure Mode and Effects Analysis, FMEA) models. In NPAS, fault models are built using a graphical programming language patterned after causal directed graphs [9]. For the PSM, the nature and significance of faults are tied to its ability to reliably distribute power to critical loads throughout the spacecraft. For this reason, a generic fault model was conceived that focuses on how a distribution path might fail to perform its function, linked upstream by failure modes that might cause that failure, and linked downstream by potentially observable events and effects of that failure, based on the electrical topology of the digital twin. At runtime, the PSM detects events and propagates up and down the links within the fault model to search for additional evidence and identify the most likely explanation for the failure (root cause isolation). The generic fault model for the PSM distribution path abstract class is shown in Figure 5.

Triggering fault diagnosis in NPAS is accomplished through a layer of event detection that is responsible for exercising the associated data analytics models and algorithms. One nice artifact of the NPAS architecture is that such event detection can readily be decoupled from the failure diagnostic logic — making it easier to test and validate each of these functions separately.

Another step in deploying an NPAS digital twin domain representation is in integrating it with data. In general, this can be done via any number of data and communications interfaces, but eventually the appropriate data must be integrated into the digital twin as sensor objects or composite object attributes. Data integration for the Artemis mission autonomy demonstration is discussed in more detail in section 6. Details on the PSM failure scenarios chosen for demonstration are provided in section 7.

A final step in integrating the PSM domain model within the PSM NPAS application is integrating the dynamic state information into a user interface that successfully conveys situation awareness to potential users. This is done through the use of proxy objects that can display state information through animation of icons. For example, a summary display of the entire PSM power system that pushes up detailed dynamic state information for improved awareness is shown in Figure 6.

5. Task and Timeline Development and Execution

The VSM's tasks and timeline execution is modeled after the NASA Gateway autonomy operations timeline and mission management framework. The layout of the framework is shown in Figure 7 and consists of the following components:

Planner: Determines the set of tasks and their order of execution that is required to transition the system from the current state to the desired goal state. The Planner sends the ordered blocks of task to the Scheduler for scheduling. Failed tasks cause the Planner to re-plan based on the current conditions.

Goal Tracker: Monitors and records the status of system's progress toward goals.

Scheduler: Receives blocks of planned tasks from the Planner and schedules them into a timeline based on timing constraints, operational constraints, system health conditions, and resource availability. The resulting timeline is segmented and sent to the Dispatcher.

Dispatcher: Manages the scheduled operational timeline of tasks. The Dispatcher is responsible for starting tasks at their scheduled time and merging new timeline segments received from the Scheduler into the operational timeline.

Task Manager: Manages the execution of tasks that the Dispatcher has started. The Task Manager uses the Constraint Evaluator (refer to Figure 7) to verify preconditions have been met, periodically checks for invariant violations, and monitors lapsed execution time for constraint violations. The Task Manager receives periodic task status updates from the Task Executor while a task is running and reacts to its

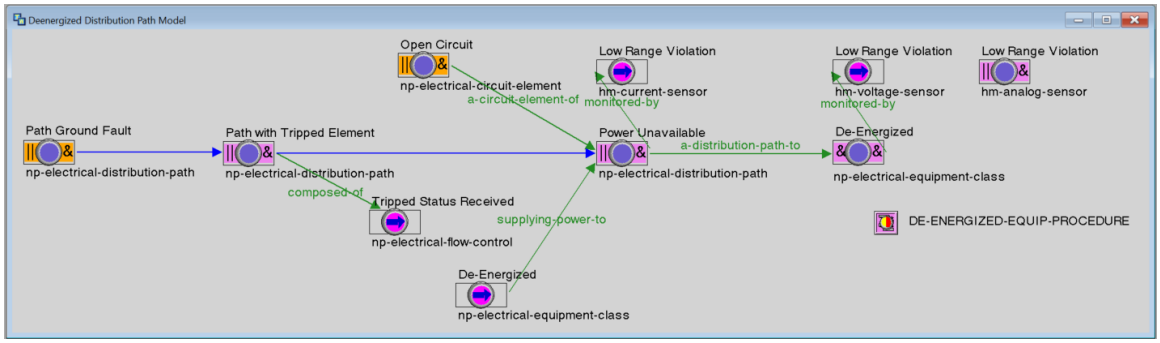

Figure 5. PSM Distribution Path Fault Model

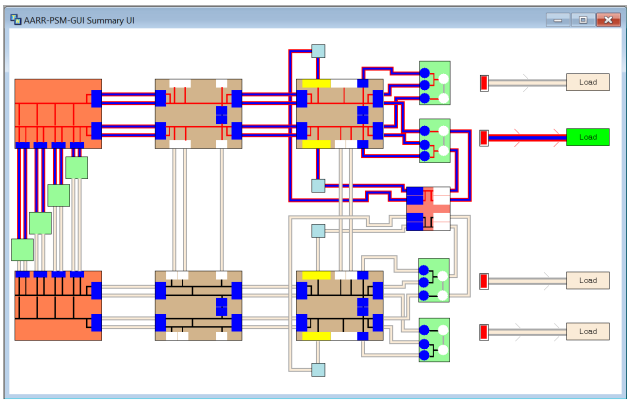

Figure 6. PSM Domain Summary Screen

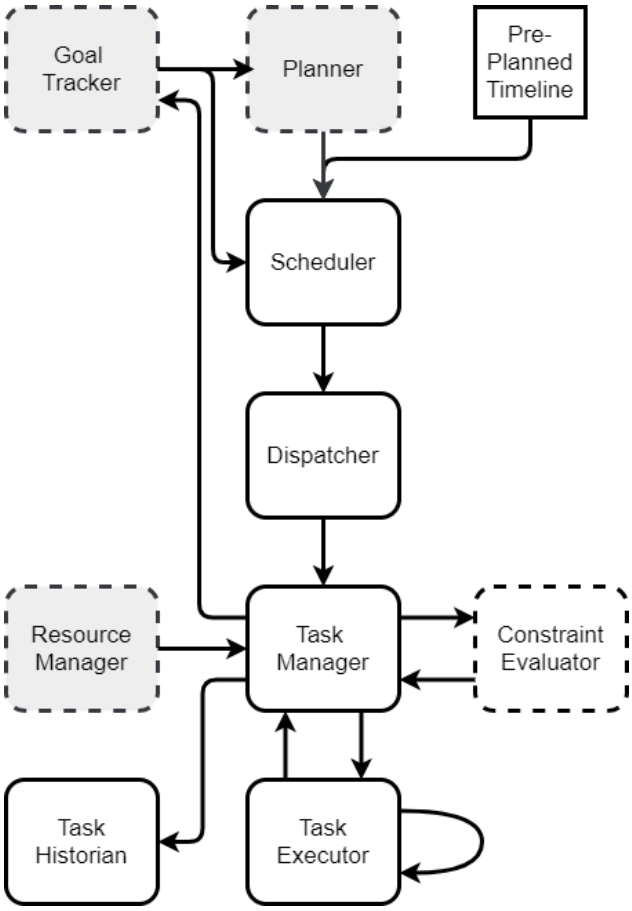

Figure 7. VSM Timeline Management and Task Execution

status — either aborting the task if a constraint is violated or notifying the Goal Tracker and Planner of success. The Task Manager always notifies the Task Historian regardless of the task's outcome.

Task Executor: Executes the task provided by the Task Manager and periodically reports the task's status to the Task Manager. A Task Executor can execute the task within its local system or forward it to lower level Autonomous System Manager. For example, a MSM would forward a power system task to its PSM.

Task Historian: Receives and records tasks as they are completed — successfully or unsuccessfully — by the Task Manager. The records serve as an audit trail for the timeline execution.

At the current stage in development, the timeline segments provided to the Scheduler are pre-defined, emulating the uploading of schedules by the ground crew to the VSM. Because of this, the Planner and Goal Tracker components have not been incorporated into this implementation. The Resource Manager and Constraint Evaluator are also rudimentary and will be improved in future releases.

6. Integration Across Modules Using SBN

All communication between VSM, MSMs, and SMs occurs across the NASA cFS Software Bus Network (SBN), an open source publish/subscribe messaging service that connects cFS instances running in separate domains [10]. In this effort, SBN was used to bridge the cFS MSM applications developed by Lockheed Martin to the NPAS VSM, MSM, and SM applications. NPAS and cFS are fundamentally different technology stacks, but they are able to communicate through

a comprehensive set of messages across the standardized SBN interface. This heterogenous architecture that emulate planned Gateway environments in which multiple vendors deliver their own ASMs that conform to the Gateway interoperability standards [5].

The NPAS applications connect to the SBN via the G2-SBN Bridge application, a commercial product developed in partnership between NASA and IgniteTech [11]. Each NPAS ASM application connects to the SBN via the G2-SBN Bridge and participates on the network as if it was a full cFS application. Figure 8 shows the block diagram of the applications and their SBN interfaces.

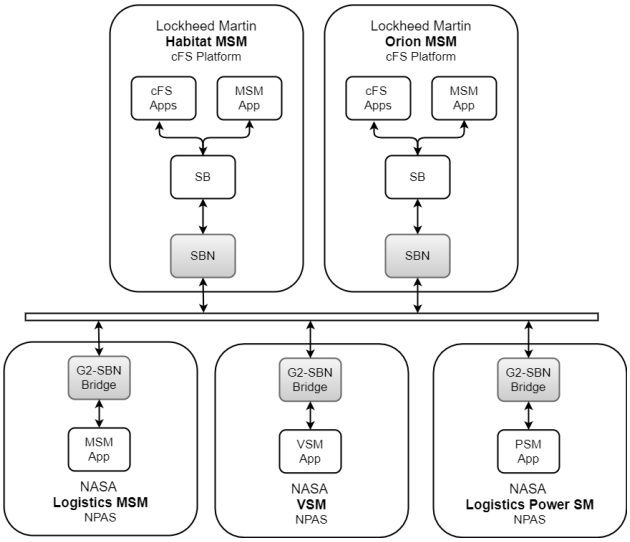

Figure 8. Network Diagram of SBN Connected Autonomous System Managers

The following SBN messages have been defined for this demonstration:

1. `summary_status_tlm` — Periodic telemetry message broadcast by all ASMs containing ID, state, and status information.
2. `activate_cmd` — Command sent by VSM to activate, restart, or register a system, device, or service.
3. `generic_response` — Reply to acknowledge most command messages. In some cases a command specific response is desired.
4. `query_model` — Request details of one or more items in another ASM's system model.
5. `query_model_response` — The reply to query_model request containing the specifics of the requested items.
6. `execute_task_cmd` — Command to create a task from a specified task template and execute the task instance with the provided parameters.
7. `task_status_tlm` — Periodic telemetry message sent by all ASMs containing the ID, state, and status of all tasks executing within an ASM.
8. `visiting_vehicle_tlm` — Notional visiting vehicle telemetry message sent by a docking spacecraft containing docking distance and fuel usage details.
9. `req_permission` — Notional 'permission to dock' message sent by visiting vehicles to the VSM.
10. `permission_response` — Reply to req_permission.
11. `psm_diagnostic_tlm` — Notional telemetry message containing the detailed results of a failure analysis performed by the PSM.

The SBN conversation that is used when an MSM first registers with the VSM — either during initial power up or as part of the docking process — is shown in Figure 9.

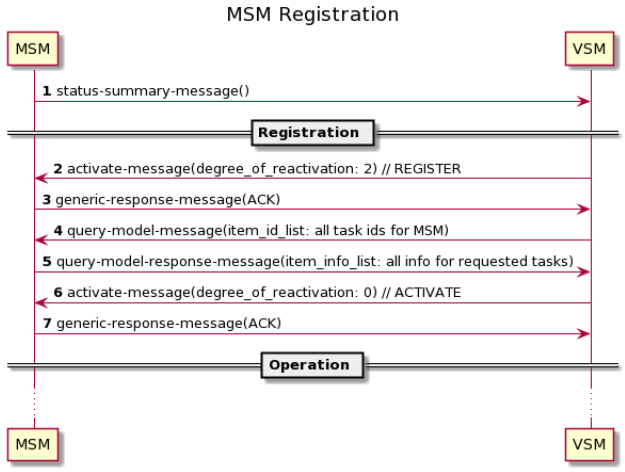

Figure 9. SBN Conversation for Visiting Vehicle Registration

The following messages are sent between VSM and MSM during a nominal registration:

1. An unregistered MSM broadcasts a status summary message with its identifying information.
2. When VSM receives a status summary message it responds with a REGISTER activation message. In parallel, the VSM loads its model of the registering MSM and the tasks it is expecting to be available.
3. MSM acknowledges the activation message.
4. VSM sends a query model message to the registering MSM, requesting the current state and status of all MSM tasks the VSM's model contains.
5. MSM replies with the current state and status of all tasks running, available to run, and not available to run. The VSM uses this information to update the Planner with the current task statuses. This information will inform the planning and scheduling of VSM and MSM goals.
6. VSM sends an ACTIVATE activation message to the MSM to command it to change its state to active.
7. MSM acknowledges the activation message to indicate the successful end of the registration process.

Once the MSM is ACTIVE it is able to participate in autonomous operations and be tasked by the VSM to help achieve the vehicle's current goals.

7. DEMONSTRATION

To highlight the capabilities of hierarchically distributed autonomous systems using VSM/MSM/SM concepts, NASA and Lockheed Martin prototyped multi-element spacecraft systems. Three use cases were designed to highlight specific aspects of the systems including: (1) registration of a Visiting Vehicle (VV) with the VSM; (2) analysis and task rescheduling by the VSM when a failure occurs in another module; and (3) Power System Manager (PSM) component failure and root cause analysis. The layout of the various modules and

system components are shown in Figure 10.

The overall system demonstrated as part of the avionics risk reduction activity was representative of a Lunar Gateway, a critical component of NASA's Artemis program. The Lunar Gateway will serve as a multi-purpose outpost orbiting the Moon that will provide essential support for long-term human return to the lunar surface and serve as a staging point for deep space exploration, and future Mars missions. For this demonstration, a VSM was in control of the overall representative Gateway system consisting of three independent modules: Habitat Module, Logistics Module and Orion as a VV, each with a respective controlling MSM. Additionally, the Logistics Module (developed by the NASA Stennis team, as noted below) included a PSM. Multiple user interfaces and a ground system were available for viewing data and interactions between the modules and the VSM.

Development of the modules and User Interface (UI) components was divided between the NASA Stennis team and Lockheed Martin. NASA was responsible for the VSM and associated UI (built with NPAS), the Logistics Module and PSM. Lockheed Martin developed the Habitat Module, Orion VV and Horizon Ground system. Orion MSM and the PSM both utilized a high-fidelity simulation developed by Lockheed Martin called SOCRRATES, which simulates Orion's electrical characteristics, sensors and vehicle dynamics, while running the vehicle flight software. Not only were these components developed by different teams, using different software platforms, but the hardware used was also geographically distributed between NASA Stennis, NASA Johnson Space Center (JSC) and a Lockheed Martin facility in Denver Colorado. The various nodes were connected through the Space Network Research Federation (SNRF) network.

Primary goals of the demonstration included the following: (1) interoperability among VSM/MSM/SM communications and interactions that enable autonomous operations; (2) design and implementation that align with the current VSM autonomy requirements and ConOps; (3) implementation of autonomy architecture down to the system manager level with VSM, MSM, and PSM with a detailed electrical schematic modeled at the power system level; (4) leverage NPAS to create schedule and execute task and timelines across multiple modules developed by separate teams; and (5) integration of Horizon Ground system for displaying telemetry and sending commands to the Habitat Module.

Use Case 1 — VSM Manages Visiting Vehicle Registration

The first use case focused on a VV docking with the Habitat module and then registering with the VSM. The docking and registration sequence begins with the Habitat and Logistics modules registered and in steady state operation. The VSM then schedules a docking task in the timeline which triggers a communication sequence with the approaching VV, and VSM granting authority to proceed. As the VV approaches, telemetry data is sent to the VSM including distance and other pertinent information. Once the docking successfully completes, the registration process begins (described in section 5).

Registration consist of notifying the VSM of the module's capabilities including task availability and resources. A sequence of steps occurs during the registration process. First, the VV begins sending a heartbeat signal to the VSM containing operational and health states. After receiving this signal, the VSM transitions to control mode which initiates a registration conversation. During the registration conversation, the VV sends its available tasks to the VSM. Using this information, the VSM intelligently adjusts the timeline and schedules new tasks. For this demonstration, a Guidance Navigation and Control (GNC) station keeping task was scheduled using the new center of mass. Using the NPAS VSM UI, an operator can view the newly available tasks and timeline, as seen in Figure 11.

The registration use case was successful in showing how registration of a new module occurs and what additional information becomes available to the overall system when a module introduces new capabilities. The VSM UI was used to view an updated task library and timeline with newly scheduled tasks.

Use Case 2 — VSM Responds to MSM Task Failure

The second use case demonstrated the actions taken by the VSM when a module reports a task failure. During normal operation, a module's MSM will run tasks as assigned by the VSM timeline. While a task is running, the MSM reports task status to the VSM at a rate of 1Hz. If the task fails for any reason, the MSM reports the failure to the VSM which updates the timeline and takes corrective actions as needed. For this use case, a task failure was manually injected to observe the system response.

Setup for this use case consisted of all modules in a registered and active state. The VSM scheduled a nominal timeline which included two tasks for the Habitat Module. These tasks were (1) maintain air quality (CO_2 removal) and (2) run science experiment (simulated data for plants removing particulate matter from the air). While these tasks were running the state is shown on the VSM UI timeline and telemetry data is streamed to the Horizon Ground System as seen in Figure 12. In addition to receiving telemetry, the Horizon Ground System is capable of sending commands. In this case it was used to inject the task failure for the running science task.

When the task failure is injected, the Habitat MSM immediately picked up the failure and updated the status information going to the VSM. When the VSM receives the new information, it marks the task as failed in the timeline and task library. In this scenario no corrective action was necessary since the failed task was a science experiment with no impact on system operation. Use case 3 delves into a component failure and how the VSM might need to adjust for a loss of a capability in one module by scheduling a new task in another module.

Use Case 3 — PSM Dynamic Health and Fault Assessment

Use Case 3 focused on demonstrating PSM fault management. This use case involved injecting symptoms (via simulated telemetry) into the power system domain model (digital twin). These telemetry measurements triggered fault event detection algorithms that have been implemented in NPAS using generic electrical system class methods and fault models (refer to Figure 5). The fault detection algorithms and fault models prompted the system to diagnose — that is, search for and identify the most likely underlying root causes based on available evidence and the knowledge obtained from FMEA.

The power system is expected to reliably distribute power across the entire spacecraft, and is therefore, designed in a modular and distributed manner which enables the ability to implement a high degree of redundancy (this ensures that all essential loads have multiple paths for receiving power).

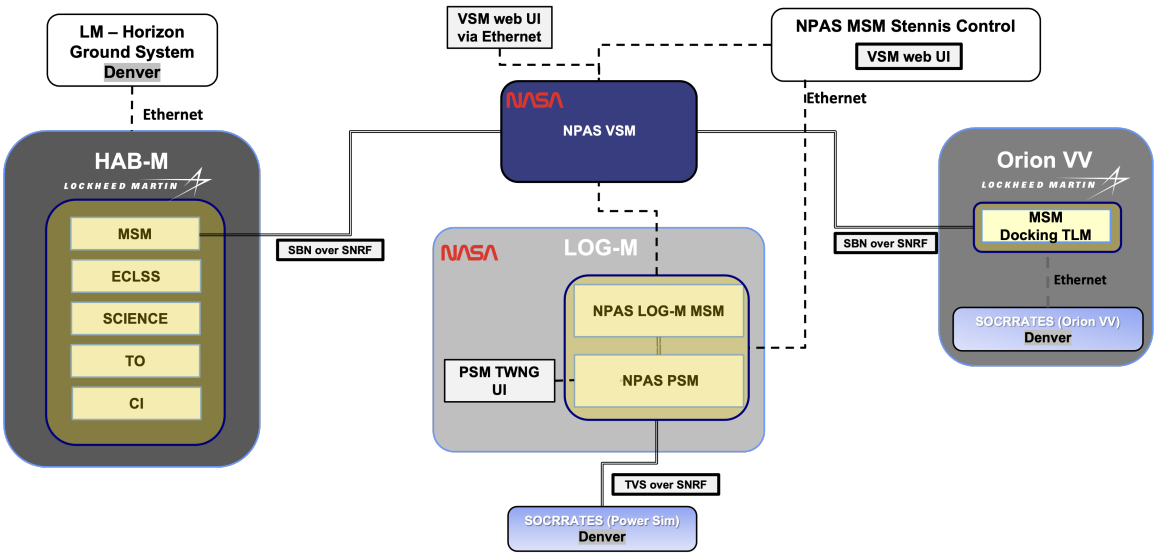

Figure 10. Demonstration Configuration

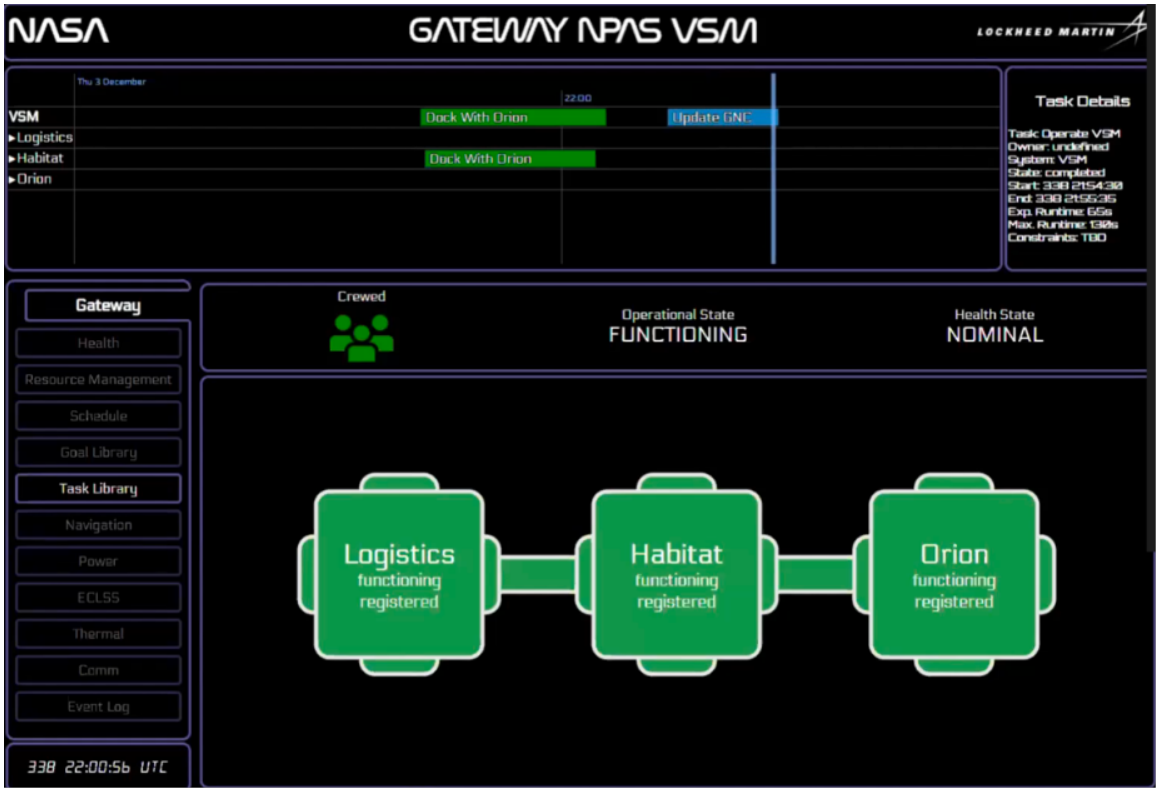

Figure 11. VSM UI

Key evidence for distinguishing and isolating root causes can therefore be gathered by reasoning over how much of the power system is being impacted by the fault. For example, when power system failures are located close to loads, only a small part of the power system should be affected. On the contrary, upstream power system faults have the tendency to impact much larger portions of the electrical distribution system (due to cascading faults). The PSM reasoner can evaluate this evidence based on topological and operational context provided by the PSM digital twin.

Furthermore, the distributed nature of the power system also readily supports the use of generic fault models. The generic fault model for electrical distribution path objects depicted in Figure 5 allows the PSM to reason over any section of the power system that is designed to distribute

Figure 12. Horizon Ground System Telemetry

power. Since the digital twin "knows"/has knowledge at any time which parts of the power system are potentially being affected by a particular component (due to dynamic traversal of the power system topology), it can use this information to readily determine the most likely location of a fault within the power system. In cases where there is insufficient data on the exact location of the fault (e.g., downstream faults in subsystems for which the PSM has no visibility), the PSM can communicate with the other SMs to help gather additional evidence.

The Use Case 3 demonstrations involved several successive simulated faults which targeted various sections of the power system, and with each successive fault, increasingly larger sections of the power system were targeted. This approach was chosen in order to validate the generic utility of the fault models, while simultaneously, based on evidence, verifying the PSMs ability to correctly identify the faulty component. An example of a diagnosed upstream fault is shown in Figure 13. Here the digital twin has assessed the faulty states of all components based on available sensor data and has performed a diagnosis. Results of the diagnosis are displayed within an internal list of inferred suspect root causes. This conclusion has been generated at run time by propagating the generic fault models based on operational and topological context, producing the run time causal-directed graph shown in Figure 14.

Another situation demonstrated as part of Use Case 3 was the ability of the PSM system to take recovery actions in response to a detected fault. Focusing on the ability of the SMs within the distributed autonomy architecture to communicate and collaborate, a fault was injected that would require the PSM to:

1. Seek additional evidence and information from another SM when insufficient evidence was available from its own telemetry stream; and
2. Prompt the VSM to request resources from another vehicle when power system faults degrade the capacity of subsystem assets.

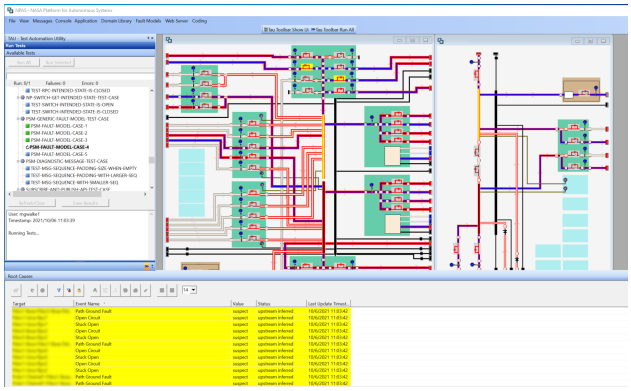

Figure 13. PSM Upstream Fault Diagnosis Example

In summary, the fault diagnosis and recovery actions performed by the PSM during Use Case 3 demonstration were all successful. When insufficient evidence was gathered, the PSM provided a correct list of suspect faults. When adequate evidence was available to isolate the failure to a specific component, the correct component was identified. Multiple locations within the power system were randomly chosen to validate the performance and utility of the generic fault models.

8. Summary and Conclusions

With this project, the team successfully demonstrated a distributed hierarchical autonomous system for a representative spacecraft comprised of a VSM, multiple MSMs, and their respective SMs. This approach aligns with the current NASA VSM autonomy design. The demonstration was driven by high-fidelity simulations to accurately represent vehicle operations. NPAS and cFS architectures were leveraged to create autonomous system managers that were used to demonstrate multi-module integration of systems developed by separate teams. Integration also included Lockheed Martin's Horizon

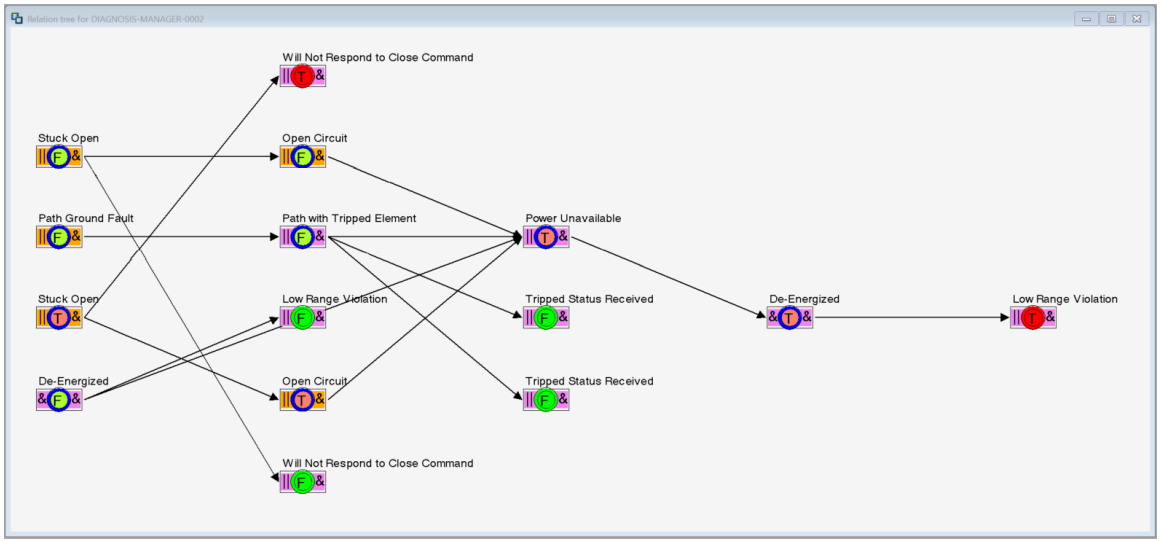

Figure 14. PSM Root Cause Isolation at Run Time

Ground system for displaying telemetry and sending commands. Three use case were developed to demonstrate various nominal and off-nominal autonomous operations. Additionally, user interfaces were created to provide situational awareness about ongoing activities.

While this demonstration for a small set of low complexity use cases was successful, a gap in concepts of operations (ConOps) for multiple complex autonomous systems is apparent. Current autonomy design defines a functional architecture consisting of multiple autonomous systems at various levels of hierarchical authority, whereby the VSM is at the top, the MSMs are below VSM, and the SMs that encompass each MSM are below the corresponding MSM. The VSM also shares the top level of the hierarchy with crew and ground support under specific operational phases and circumstances. The gap is the ConOps for this autonomy design, whereby the expectation is that each autonomous system is highly capable; this concept is just now being developed. The complexity of this ConOps is well beyond ConOps for autonomous activities, not autonomous systems, in recent NASA experience (e.g., navigation on Mars surface, docking on ISS) in terms of the number and complexity of the autonomous systems.

ACKNOWLEDGMENTS

The authors would like to acknowledge support from NASA Advanced Exploration Systems (AES), that made possible the development and implementations of NPAS. The authors would like to thank Danny Carrejo, JSC Exploration Integration and Test Lead for facilitating the support at JSC. The authors would like to acknowledge Chris. Moore, NASA AES, Exploration Capabilities Manager and Denise Varga, AES Habitations Support Systems Domain Lead for HQ guidance and support. C. Duane Armstrong, NASA SSC, Test Technology Branch Chief, for additional guidance and support. The authors would also like to acknowledge the technical proficiency and support from D2K Technologies, Quentin Oswald, Joshua Broberg, Brian Rey, Federico Piatti, and Michael Walker whose expertise enabled the capabilities presented in this paper. The authors would also like to acknowledge the technical expertise provided by Lockheed Martin developers Mike Rosenberg, Cory Kennedy and Jaret Anderson. The authors would like to acknowledge the Lockheed Martin Horizon team including Al Faymore, Christian Morrow, and John Slezosky who were instrumental in integrating the Habitat module with a ground command and control system. Lastly, the authors would like to acknowledge Angie Kibler and Matt Goman, Lockheed Martin, who provided essential support and expertise.

REFERENCES

[1] Technology Roadmaps, TA 4: Robotics and Autonomous Systems, 2015, www.nasa.gov

[2] Fernando Figueroa, Mark Walker, Lauren Underwood, and Jonathan Morris, "NASA Platform for Autonomous Systems (NPAS)," AIAA SciTech 2019 Forum, San Diego, California, AIAA 2019-1963.

[3] Fernando Figueroa and Mark Walker, "Integrated System Health Management (ISHM) and Autonomy," AIAA SciTech Forum 2018, 8-12 January 2018, Kissimmee, Florida, AIAA 2018-1152

[4] Jason Crusan et al., "Deep space gateway concept: Extending human presence into cislunar space", IEEE Aerospace 2018, Big Sky, Montana DOI: 10.1109/AERO.2018.8396541

[5] Jason Crusan et al., "NASA's Gateway: An Update on Progress and Plans for Extending Human Presence to Cislunar Space", IEEE Aerospace 2019, Big Sky Montana, DOI: 10.1109/AERO.2019.8741561

[6] Julia M. Badger, Phillip Strawser and Charles Claunch, "A Distributed Hierarchical Framework for Autonomous Spacecraft Control, IEEE Aerospace 2019, Big Sky, Montana, DOI: 10.1109/AERO.2019.8742199

[7] Fernando Figueroa, Lauren Underwood, Ben Hekman and Jonathan Morris, "Hierarchical Distributed Autonomy: Implementation Platform and Processes", 2020 IEEE Aerospace Conference, 2020, pp. 1-9, doi: 10.1109/AERO47225.2020.9172572

[8] Julia Badger, Charles Claunch, and Frank Mathis, "Modular Autonomous Systems Technology Framework: A Distributed Solution for System Monitoring and Control," ntrs.nasa.gov

[9] R. Kapadia. "SymCure: A model-based approach for fault management with causal directed graphs", Proc. Of the 16th Intl. Conf. IEA/AIE-03, pp. 582-591, Loughborough, UK

[10] "Core Flight Software (cFS) Software Bus Networking (SBN)," NASA, [Online]. Available: github.com/nasa. [Accessed 1 October 2021]

[11] "IgniteTech," [Online]. Available: ignitetech.com. [Accessed 1 October 2021]

[12] "cFE Users Guide" [Online]. Available: cFE_Users_Guide.pdf. [Accessed 1 October 2021]

BIOGRAPHY

Fernando Figueroa *received his BS Degree at the University of Hartford, CT (1983), and MS and Ph.D. Degrees at The Pennsylvania State University, PA (1984, 1988); all in Mechanical Engineering. He was faculty of Mechanical Engineering at Tulane University (New Orleans) for 10 years, and Associate Chair of Advanced Instrumentation and Control at The University of New Brunswick (Canada) for 2 years. He has been at NASA Stennis Space Center since 2000. He has led multiple R&D projects in collaboration with academia, industry, government agencies, and other NASA centers. He is currently Lead for Autonomous Systems and Operations, working on projects funded by NASA Advanced Exploration Systems, Space Technology Mission Directorate, and NASA Stennis Space Center. His areas of interest include Autonomous Operations and Systems, Integrated System Health Management (ISHM), Intelligent Systems, Intelligent Sensors, Robotics, and Automatic Controls.*

Lauren Underwood *received her BS, MS, and PhD Degrees in Biology (developmental neurobiology with focus in optics) from Tulane University, LA (1985, 1987 & 1991). Following graduation, she was awarded a National Institute of Health Training Grant Fellowship in Vision Research for post-doctoral studies (1991-1993). Dr. Underwood has been at NASA Stennis Space Center (SSC) since 2001, and has worked directly for the Office of the Chief Technologist (OCT), as well as supported activities associated with Science Mission Directorate, Science Technology Mission Directorate (STMD), DEVELOP and most recently works in the Test Technology Branch in the Engineering and Test Directorate at SSC. Currently, she serves as a SSC Project Manager, for NASA's Advance Exploration Systems (AES) autonomy software activities, as well as serves the Autonomous Systems Lab project manager for other related autonomy software infusion activities at SSC.*

Mark G. Walker *received his BSEE from California Polytechnic University, Pomona (1990), and his MSCompEng from the University of Southern California, Los Angeles, CA (1994), where he specialized in machine intelligence. His experience in artificial intelligence began in 1989 as a DOE undergraduate fellow at Oak Ridge National Laboratory where he developed image processing and perception software for autonomous robots. His work with HUMS and ISHM began with BFGoodrich Aerospace, Vergennes, VT in 1996, where he delivered AI solutions for the F-35 program and co-authored four patents in the field. He also spent 6 years as Senior Consulting Engineer for expert system manufacturer Gensym Corporation and 10 years as Lead Engineer, Intelligent Systems for General Atomics, where he led GA in the development of reusable PHM systems applied to various industries. He founded D2K Technologies in 2014, a solution provider of intelligent model-based reasoning solutions for mission critical systems. He also serves as an ISHM and Autonomous Operations SME for NASA, with active projects at SSC. He resides with his family in Oceanside, California.*

Jonathan Morris *received his BS Degree in Electrical and Computer Engineering from Rowan University, NJ (2003). He has worked with Stennis Space Center for more than 15 years to develop and integrate new technologies into NASA's operations — with a focus on measurements and integrated system health management and intelligent systems. He is currently the Vice President of Engineering at D2K Technologies where he is fortunate to work with a team of dedicated and bright engineers.*

Rane Brown *received his BS Degree in Electrical and Computer Engineering from the University of Colorado Boulder, CO (2016). He has worked at Lockheed Martin Space for five years with a most recent role in the Human Spaceflight Exploration division of Advanced Programs. In that role he worked closely with NASA Stennis Space Center developing software for control and simulation of autonomous systems. Other experiences include development of automated test software and embedded flight software for a range of spacecraft bus types.*