https://ntrs.nasa.gov/search.jsp?R=20190002469 2020-05-09T21:58:41+00:00Z

A Distributed Hierarchical Framework for Autonomous Spacecraft Control

Julia M. Badger, Philip Strawser
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
281-483-2277
{julia.m.badger, philip.a.strawser}@nasa.gov

Charles Claunch
GeoControl Systems
2900 Woodridge Dr. Suite 100
Houston, TX 77087
281-483-1580
charles.a.claunch@nasa.gov

Abstract— **Future human space missions for exploring beyond low Earth orbit are in the conceptual design stage. One such mission describes a habitat in cis-lunar orbit that is visited by crew periodically, others describe missions to Mars. These missions have one important thing in common: the need for autonomy on the spacecraft. This need stems from the latency and bandwidth constraints on communications between the vehicle and ground control. A variable amount of autonomy may be necessary whether the spacecraft has crew on board or not.**

Spacecraft are complex systems that are engineered as a collection of subsystems. These subsystems work together to control the overall state of the spacecraft. As such, solutions that increase the autonomy of the spacecraft (called autonomous functions) should respect both the independence and interconnectedness of the spacecraft subsystems. This distributed and hierarchical approach to system monitoring and control is a key idea in the Modular Autonomous Systems Technology (MAST) framework.

The MAST framework enables a component-based architecture that provides interfaces and structure to developing autonomous technologies. The framework enforces a distributed, hierarchical architecture for autonomous control systems across subsystems, systems, elements, and vehicles. An example autonomous system was implemented in this framework and tested using realistic spacecraft software and hardware simulations. This paper will discuss the framework, tests conducted, results, and future work.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. BACKGROUND INFORMATION 2
3. THE MAST FRAMEWORK 3
4. EXPERIMENTS 5
5. CONCLUSIONS AND FUTURE WORK 5
ACKNOWLEDGEMENTS 6
REFERENCES 6
BIOGRAPHY 7

1. INTRODUCTION

Future exploration missions that will send humans beyond near Earth orbit are in the planning stages at NASA. A common concept of operations for these missions is to emplace habitats, spacecraft, and logistics in advance of the arrival of the crew. This important equipment will remain in place between crewed missions, but during this time, it is essential that the health of these assets is maintained. Ground operations support will clearly play a role in this, but with reduced communication bandwidth and increased latency, operations must advance beyond the paradigm of the International Space Station (ISS). As such, research into what technologies are needed to enable the autonomous operation of not always crewed human spacecraft is underway.

A key concept in this work is vehicle systems management. This paradigm assumes that cross-system or vehicle level decisions will need to be made while out of communication contact with ground control. There are two important methods of vehicle systems management. The first is necessary when crew is on board and operating the vehicle, but requires support that the ground controllers cannot give. The second involves vehicle control when no one is on board. These uncrewed scenarios occur frequently in advanced mission concepts. The Gateway [1], a cis-lunar habitat that will serve as the access point to lunar and Martian destinations, is expected to be uninhabited for 11 months per year, and for up to 3 years at a time.

Although Gateway is not the bounding case in terms of technical difficulty, since communication into cis-lunar space is expected to be frequent and to have low latency, it does have its challenges. The Gateway vehicle will be comprised of several modules that will be built by various space agencies around the globe. Like the International Space Station, these modules will require tight integration for vehicle control. For example, the life support systems will be present on some, but not all, habitable modules, and the life support systems that exist will have to function appropriately as redundant capabilities for the vehicle stack. This collection of subsystems and modules are both interconnected and independent, which is a recipe for operational complexity. Unfortunately, the management of operational complexity is largely out of reach for most autonomous systems technologies today.

This paper will detail the efforts to develop a framework that would be capable of the successful operation of a complex human spacecraft, while respecting the independence and interconnectedness of its components and subsystems. The Modular Autonomous Systems Technology (MAST)

framework enables a component-based architecture that provides interfaces and structure to developing autonomous technologies. The framework enforces a distributed, hierarchical architecture for autonomous control systems across subsystems, systems, elements, and vehicles. The framework supports communication and transparent interfaces between its components and enforces a strict command and telemetry flow as a systems engineering tool. The most unique part of this framework is the inclusion of contract based design concepts that encourages design for verification methodologies and supports component-level verification playing an important role in overall system verification.

This paper is organized as follows. Section 2 will give some background on previous work on vehicle system management and associated frameworks. Section 3 will describe the MAST framework itself. Section 4 provide details on testing that MAST has undergone and results of these tests. Section 5 will conclude the paper with a focus on the vision of future work along this promising path.

2. BACKGROUND INFORMATION

Two important studies were conducted into the autonomous operation of periodically crewed human spacecraft [2, 3]. The first study defined dormancy as uncrewed flight that featured a reduced set of operations and described the mission stages of uncrewed operations, phases of dormant operations, and critical system capabilities that are needed for dormant operations. This study provided a brief comparison of dormancy operations of past robotic missions to identify lessons that can be applied to planned human exploration missions. The subsequent study in [3] provided a deep-dive analysis into dormant operations on a subsystem basis. The analysis compared the state of the art in human spacecraft operation (ISS) with the requirements that will drive the operation of an uncrewed human spacecraft in Martian orbit. The resulting technology gaps were assessed and recommendations for future development were described. One of the main recommendations for the control of uncrewed and dormant spacecraft was the inclusion of a vehicle systems manager (VSM) to provide integrated, vehicle-level command and control of the spacecraft.

Previous work is mostly found in technologies that contribute to a VSM-like function. While the ISS was not designed for dormant operations, numerous innovations in autonomous payload and core systems control and monitoring have been made. A summary of these advances [4] includes data downlink of accelerometer data, onboard thermal management, onboard data bandwidth management, scientific payload cold storage monitoring and operations, and power systems monitoring and emergency response. Though these technologies have made promising progress towards realizing autonomous systems management, these capabilities were developed for an active crewed spacecraft managed from Earth. As such, they do not represent an integrated vehicle systems management solution.

Advanced research has developed a complete fault management capability referred to as Advanced Caution and Warning System (ACAWS) [5]. ACAWS splits the fault management task into fault detection, fault isolation, and fault impacts reasoning, but uses a single spacecraft component and fault model. First tested on a low-fidelity surface habitat, ACAWS is being adapted to perform fault management for the Orion spacecraft, both for flight controllers and also for crew [6]. The Autonomous Power Controller subsystem level controller and fault management functions are integrated with a VSM including a spacecraft-wide automated planner, subsystem level fault management, and plan execution system, running on modern avionics and path-to-flight hardware [7]. While this work demonstrates the successful application of a hierarchical autonomous system architecture, the scope of the experiment was limited and did not demonstrate subsystem integration or interconnectedness. Likewise, plan execution technology has been tested onboard the ISS as a way to automate payload operations. AMO EXPRESS [8] describes a demonstration of how an experiment facility can be autonomously operated, with simple integrated fault detection and response capabilities. The NASA Platform for Autonomous Systems (NPAS), is a software platform used to make systems operate autonomously using a model-based systems engineering (MBSE) approach [9]. NPAS is able to use live models for real-time autonomous operations, largely for integrated system health management.

Plant operations, such as water processing [10], have benefited from similar systems management technologies. These examples integrate many processes and subsystems into decision support and autonomous operations tools. However, these systems (though larger) are typically less complex and less interconnected than a human spacecraft's subsystems. Likewise, a sort of vehicle systems management occurs on robotic spacecraft en route to deep space destinations, but as noted in [1], these systems typically employ their long time to effect to bring the spacecraft to a safe state for ground controllers to assess and recover. Human spacecraft will not have the same recovery options in many circumstances.

Autonomous systems are complex, difficult to test, and nearly impossible to conduct formal analysis on to find performance guarantees. However, the use of autonomous systems technology for human spacecraft will require convincing verification and validation. The MAST framework has a path to formal analysis and will create assume-guarantee contracts as long as the autonomous technology components can be verified individually. This paper will describe the successful integration of several subsystems, modules, and processes with a vehicle system manager in the MAST framework and discuss the contract-based design approach that was taken.

3. THE MAST FRAMEWORK

The MAST framework is a component-based system that provides interfaces and structure to developing autonomous technologies. The categories of technologies are broken into several "buckets" (see Figure 1) that are based on the OODA loop[1] (Observe, Orient, Decide, Act) concept. These buckets are each identified with an autonomous functionality that is needed in the control of an autonomous system. There are three types of autonomous systems that will be defined:

1. Spacecraft subsystem - operates independently both nominally and in response to fault detection, isolation and recovery; examples are Power, Communications, Life Support.
2. Mechanical events & processes – examples include docking of spacecraft (i.e., Automated Rendezvous and Docking), grappling with robotic manipulators.
3. System-level Intelligence – onboard ability for system-level planning, health monitoring, and mission management; example is the Vehicle System Manager (VSM).

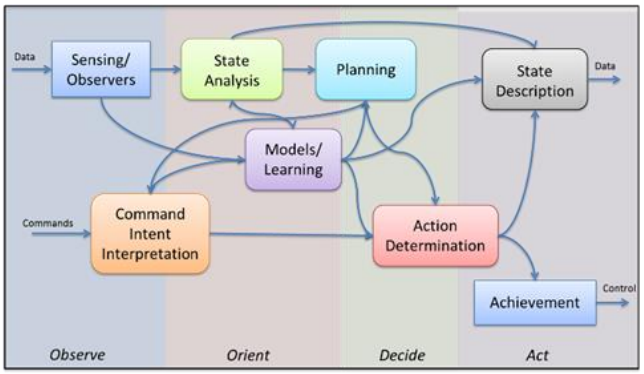

Figure 1: Open-loop Framework Diagram

These various types of autonomous systems that will be implemented with buckets of autonomous functionality are henceforth referred to as "clusters." Each cluster will have 0 to n buckets of each type, depending on the needs of the system that the cluster is servicing. The various buckets will have different requirements and structure, but this section will first expound upon three main reasons for creating this architecture:

1. Using products from autonomy across levels of abstraction,
2. Creating systems that are straight-forward to verify, or are constructed with guarantees, and
3. Allowing for variable autonomy.

Figure 2 gives an illustration of an example spacecraft architecture that has several autonomous modules, where each autonomous module is associated with a cluster, which contains an instance of the component-based architecture shown in the Figure 1 above. This example architecture is loosely based on the Gateway concept of Autonomous Systems Management (ASM) architecture.

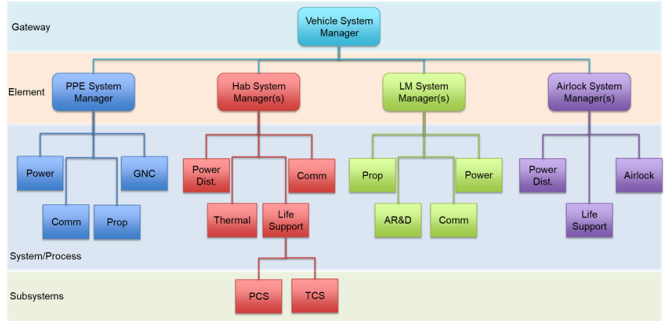

Figure 2: Example Autonomous Spacecraft Diagram

Distributed and Hierarchical Architecture

A key component of this framework is its ability to support a distributed and hierarchical architecture. This architecture is a common systems engineering construct used to reduce overall complexity by allowing components of the system to handle what they can and delegate up issues that are outside of their purview. The MAST framework supports this architecture by providing templates for command and telemetry flow through this architecture. For example, the State Description bucket provides for a unified message to send telemetry and requests up the hierarchy; at the VSM level, this State Description message would provide information flow to the human operators (on board or in ground control). The Achievement block would send commands down to clusters on a lower level of the hierarchy. Likewise, the Sensors/Observers block would accept data from hardware or from the State Description message from lower level clusters and the Command Intent Interpretation bucket would accept command messages from either ground control (VSM) or clusters above it.

An example of this data flow is as follows. Consider a trip on a circuit in the Power Distribution (PD) subsystem that removes power from the Life Support (LS) system's Spacecraft Atmosphere Monitor (SAM) and some payloads. The PD autonomous cluster would sense this fault and send information up to the Habitat Element System Manager (HESM). The LS cluster would likewise sense that the SAM had been disrupted, but would know that the cause could be internal (fault in the SAM) or external (power loss) to the LS cluster. Therefore, it would send information up to the HESM as well. The HESM cluster, with this data, would be able to instruct the LS cluster to standby with respect to this error while the PD subsystem generated and executed recovery options.

This example is simple, but it makes important points while allowing for several quick extensions. For example, assume that the trip was due to overcurrent caused by an error in one of the payloads on the circuit. This would drive the inclusion of a Payload Systems Manager (PSM) that accumulates the

[1] https://en.wikipedia.org/wiki/OODA_loop

states of the payloads. It would also mean that initial recovery options by the PD cluster would be unsuccessful. At this point, the HESM would have to get involved with recovery, for example, and choose to turn off the payload (either due to priority or due to data from the PSM) before commanding the PD to again reset the circuit. One could also imagine that the power problem somehow originated from power creation, which for Gateway, resides on a different module. In this case, the diagnosis and recovery process would flow up to the VSM cluster as well.

One of the pre-requisites for a distributed and hierarchical fault response as outlined above is the ability to have models that support consistent levels of abstraction. Another pre-requisite is well-defined interfaces between systems on the same level and between clusters on the lower level. What this means is that each cluster needs to know when it depends on a different cluster. Going back to the example, the LS cluster knows that it cannot diagnose the SAM failure due to the dependency of the fault tree on data from the PD system. Instead of having each cluster handle the acquisition of this data from the appropriate cluster, the framework stipulates that data connections can only be made by the cluster one level up in the hierarchy. In that sense, the LS model knows about the dependency, but the HESM model knows what the dependency is. This means that the level of abstraction of each cluster's model fits to its purpose and level on the hierarchy.

Further requirements on data sharing and model consistency include the following:

- The framework shall enforce consistency of model definition.
- The variables in the models shall self-enforce units and assumptions (units and assumptions should be explicit in variable definition).
- MAST shall ensure visibility and query-ability of variables and products within hierarchical constraints as a rule (truly internal variables should be discouraged).

Design for Verification

Autonomous systems are complex, difficult to test, and nearly impossible to conduct formal analysis with guarantees. However, the use of autonomous systems technology for human spacecraft will require convincing validation and verification; for systems with emergent behaviors, this requirement becomes even further out of reach of the state-of-the-art. The MAST framework has been built with a path to formal analysis, and allows the designer the potential of creating guarantees as long as the autonomous technology buckets can be verified individually. Specific requirements include the following:

- The framework shall have the ability to interface with temporal logic specifications.
- The framework components shall require specific definitions for the incoming and outgoing data.

Thresholds could be defined as part of the dataports, for example, power data input can only be from 0-100. Errors would be thrown if data were out of range.

Specifically, the MAST framework supports a contract-based design approach [11]. The contract-based design can be implemented on several levels, but the framework right now enforces contracts within the cluster, between the buckets. This is instantiated in the following way. First, each bucket supports having a set of assumptions on the data that comes into the bucket. The assumptions that can be expressed as simple logical expressions can be checked in real-time as data enters the bucket. Likewise, buckets support guarantees on data exiting the bucket. The guarantees that can be expressed as simple logical expressions can be checked in real-time just before data exits the bucket. These assume-guarantee contracts between the buckets can be verified using various formal methods techniques. This approach provides a benefit in that the buckets themselves then only have to be verified as satisfying the guarantees, given the assumptions.

The checks on the assumptions and guarantees on each bucket can be entered into a YAML[2] file for that bucket. The MAST framework supports reading in this configuration file at runtime and will automatically run the checks at the appropriate times. These checks can be tied, via the configuration file, to separate callbacks for successful or failing checks. These callbacks can be used to disrupt the flow of execution of the bucket, if necessary. For example, if an incoming (assumption) check fails, the bucket could not possibly run as intended. As such, the execution could simply fail with a message, giving operators an indication of where the failure originates. Alternatively, the callback could check a broader set of assumptions, and execution of the bucket could continue using an alternate control sequence that satisfies only a subset of the guarantees. This behavior is important if guarantees include both safety and performance specifications. Upon failures, safety specifications could be maintained while performance guarantees are sacrificed.

A similar interaction with the outgoing checks can occur, but the difference here is that the option to return execution to the bucket is given. This gives the bucket the chance to self-correct upon guarantee failure, for similar reasons as given above.

Variable Autonomy

Because the ASM architecture is meant to be used with human spacecraft that will see both crewed and uncrewed stages, there is a range of autonomy that will be required for operation. For example, the communications system may need to be fully autonomous during dormancy, but can be crew-controlled during critical stages in Mars orbit insertion.

[2] http://yaml.org/

A key assumption for this feature is that the "reasoning" part of the autonomous system will not need to be variable- there should always be data analysis, planning, and state description. However, the important parts of the system to have an "autonomy dial" are the command and action-based components. So, requirements for this feature are given more on a component-by-component basis.

Additional Features

The MAST framework has been designed for distributed execution to support the ASM architecture. This is implemented through its integration with Core Flight Software (cFS)[3]. MAST allows the application to be split along cluster lines. All of the buckets in the cluster (running the autonomous control loop) must run in the same process. The inter-cluster communication uses a “blackboard” that allows quick data transfer between the buckets in the cluster. It also facilitates the minimization of check occurrences when possible.

The MAST framework facilitates data logging via its integration with the Lightweight Accumulator Gathering Efficiently in Real-time (LAGER) logging software. LAGER supports zero-copy transport and minimal code interfaces. It features efficient file writing/sizing and is built for various data sources (taps), accumulators (kegs), and consumers (mugs). Figure 3 shows a representation of the LAGER software.

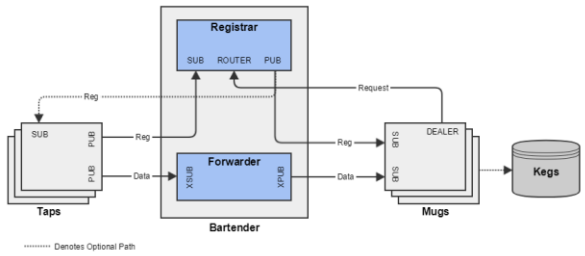

Figure 3: LAGER

Trend analysis is an important function that many types of autonomous systems require. The integration with LAGER supports the on-line creation of trending baselines and trend determination. Likewise, mode and resource management are important functions for autonomous systems, and libraries incorporating these capabilities have been integrated into MAST as well.

4. Experiments

Two experimental scenarios were implemented and run within the MAST framework. The first follows from the power fault scenario previously described. In addition to the interactions between the power distribution and life support systems, the scenario also included an Automated Rendezvous and Docking (ARD) process with Orion. This autonomous process featured a flight rule that the rendezvous would be paused at certain hold points if the atmosphere inside the habitat was unacceptable. As such, the VSM component was able to pause the ARD process and direct the recovery of the overcurrent PD fault by turning off a payload. Once the SAM recovered, the ARD process was commanded to continue.

The second experiment involved failures that were more continuous in nature. The scenario involved a slow coolant loop leak into the cabin. The extra water increased the humidity of the atmosphere, which would drive the system to slowly increase the temperature of the cabin to accommodate it. During an eclipse, this stress on the shell heaters could uncover battery cell degradation. This experiment featured two trend analyzers, for coolant level and battery power draw. The scenario also exercised the command and control architecture by adding element system managers to the ASM implementation, shown in Figure 4 below.

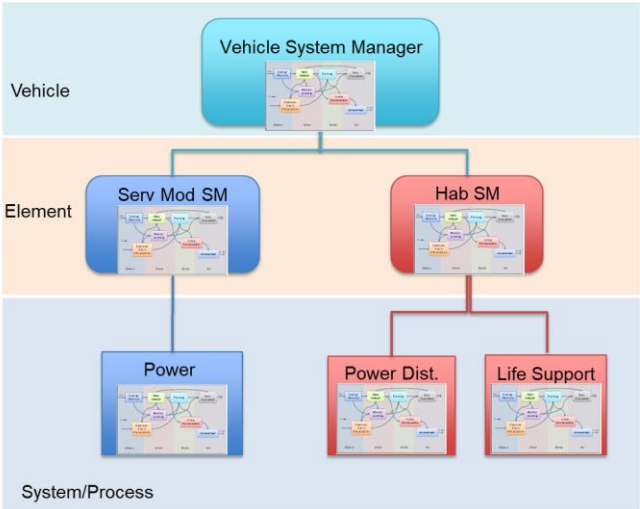

Figure 4: ASM Architecture for Second Experiment

In both experiments, the MAST framework worked as expected. The benefits of the contract checks were immediately useful, as these checks were capable of finding errors during the development and integration of the autonomous system quickly and efficiently.

5. Conclusions and Future Work

A framework to support the operation of a distributed, interconnected system of systems, such as human spacecraft, was presented. The MAST framework supports the careful design of interconnections between distributed systems through a hierarchical command and control architecture. This framework promotes a design for verification paradigm through the integration of runtime monitoring and contract-based design. MAST has been applied to example scenarios that incorporate several spacecraft subsystems and processes to demonstrate feasibility and performance.

Future directions for the MAST framework are many. First, tighter integration with spacecraft subsystems will require the

[3] https://cfs.gsfc.nasa.gov/

full adoption of cFS. This will increase the performance and require more stringent integration with real-time operating systems. Second, technology integration to support the planning and execution pipeline is needed to close the loop in the MAST cluster. Current technologies rarely support the type of distributed and hierarchical control needed for the proposed Gateway control architecture, and so this is another future direction.

Data management is hugely important in spacecraft control, particularly because it is a resource with availability constraints not commonly encountered on Earth. Loss of communications, latency, and reduced bandwidth as humans travel further from Earth are all challenges to which any autonomous spacecraft must be robust. The MAST framework needs to have utilities available to support these realities. Finally, the integration of other systems and system managers will continue to stress the framework. For example, the addition of a robotic spacecraft inspector or maintainer will provide the framework another resource to schedule and more recovery options for various faults and emergencies.

The MAST framework and associated experiments have provided inspiration for the direction of the Gateway Vehicle Systems Manager and autonomy architecture. MAST will continue to play a role in the technology development and requirements creation leading up to the next destination in human space travel and beyond.

ACKNOWLEDGEMENTS

The authors thank Patrick Knauth, William Othon, Daniel Carrejo, Zach Crues, Paul Bielski, Zu Qun Li, Jason Harvey, and Miriam Sargusingh for their essential contributions to this project.

REFERENCES

[1] Crusan, J.C., Smith, R.M., Craig, D.A., Caram, J.M., Guidi, J., Gates, M., Krezel, J.M., and Herrmann, N. "Deep Space Gateway Concept: Extending Human Presence into Cislunar Space." Proceedings of the IEEE Aerospace Conference, 2018.

[2] Williams-Byrd, J., Antol, J., Jefferies, S., Goodliff, K., Williams, P., Ambrose, R., Sylvester, A., Anderson, M., Dinsmore, C., Hoffman, S., Lawrence, J., Seibert, M., Schier, J., Frank, J., Alexander, L., Ruff, G., Soeder, J., Guinn, J., and Stafford, M. "Design Considerations for Spacecraft Operations During Uncrewed Dormant Phases of Human Exploration Missions." Proceedings of the International Astronautical Congress, 2016.

[3] Badger, J., et al. "Spacecraft Dormancy Autonomy Analysis for a Crewed Martian Mission." NASA/TM-2018-219965, 2018.

[4] Cornelius, R. and Frank, J. "International Space Station (ISS) Payload Autonomous Operations Past, Present and Future." Proceedings of the AIAA Conference on Space Operations, 2016.

[5] McCann, R., Spirkovska, L., and Smith, I. "Putting ISHM Capabilities to Work: Development of an Advanced Caution and Warning System for Crewed Spacecraft." Proceedings of the AIAA Modeling and Simulation Technologies Conference, 2013.

[6] Aaseng, G., Barszcz, E., Valdez, H., and Moses, H. "Scaling Up Model-Based Diagnostic and Fault Effects Reasoning for Spacecraft." Proceedings of the AIAA Conference on Space Operations, 2015.

[7] Aaseng, G., Frank, J., Iatauro, M., Knight, C., Levinson, R., Ossenfort, J., Scott, M., Sweet, A., Csank, J., Soeder, J., Loveless, A., Carrejo, D., Ngo, T., and Greenwood, Z. "Development and Testing of a Vehicle Management System for Autonomous Spacecraft Habitat Operations." Proceedings of the AIAA Space Conference, 2018.

[8] Stetson, H., Frank, J., Haddock, A., Cornelius, R., Wang, L., and Garner, L. "AMO EXPRESS: A Command and Control Experiment for Crew Autonomy." Proceedings of the AIAA Conference on Space Operations, September 2015.

[9] Walker, M., Figueroa. F. and Toro-Medina, J., "PHM enabled autonomous propellant loading operations," 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-11.

[10] Stein, D., Achari, G., Langford, C. H., Dore, M. H., Haider, H., Zhang, K. and Sadiq, R., "Performance management of small water treatment plant operations: a decision support system." Water and Environment Journal, 31: 330-344. 2017.

[11] P. Nuzzo, M. Lora, Y. A. Feldman and A. L. Sangiovanni-Vincentelli, "CHASE: Contract-based requirement engineering for cyber-physical system design," 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2018, pp. 839-844.

BIOGRAPHY

Dr. Julia Badger *is the Project Manager for the Robotics and Intelligence for Human Spacecraft team at NASA-Johnson Space Center. She is responsible for the research and development of humanoid robotic (Robonaut) and autonomous system capabilities, on the Earth, the International Space Station, and for future exploration, that include dexterous manipulation, autonomous spacecraft control and caretaking, and human-robot interfaces. Julia has a BS from Purdue University, and an MS and PhD from the California Institute of Technology, all in Mechanical Engineering. Her work has been honored with several awards, including NASA Software of the Year, Early Career, and Director's Commendation Awards.*

Philip Strawser *received a B.S. in Computer Engineering at Georgia Tech in 2002. In 2002, he joined NASA's Johnson Space Center (JSC) in Houston, Texas. At NASA, Mr. Strawser works in the Robotic Systems Technology branch of the Software, Robotics, and Simulation division. Initially focused on avionics and embedded systems, he helped to design and implement several robotic platforms including Robonaut, Spidernaut, and Centaur. He later focused on software development, and in 2007, he led the software team to develop Robonaut 2. He led the software certification effort which allowed R2 to go to the International Space Station in 2011. In 2012, Mr. Strawser worked with a JSC team to design and develop the Valkyrie robot for the DARPA Robotics Challenge. Mr. Strawser is currently the Perception and Cognition lead for the Robonaut 2 system. His interests are in computer vision, machine learning, and robotic task design and execution.*

Charles "Chuck" Claunch *is a software engineer with expertise in systems integration and software design. He has a B.S. in Computer Science from Texas Tech University and worked in the telecom industry for a few years before his over ten years at NASA. At NASA, Chuck has worked on several projects that have flown to the International Space Station.*