Assurance of Model-Based Fault Diagnosis

Allen Nikora, Priyanka Srivastava, Lorraine Fesq, Seung Chung
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
{818} 393-1104, 354-7567, 354-7014, 393-7224
{Allen.P.Nikora, Priyanka.Srivastava, Lorraine.M.Fesq, Seung.H.Chung}@jpl.nasa.gov

Ksenia Kolcio
Okean Solutions, Inc.
1463 East Republican Street, 32A
Seattle, WA 98112
310-704-6174
ksenia@okeansolutions.com

Abstract— **Autonomy is an increasingly important technology for robotic scientific and commercial spacecraft. An important motivation for developing onboard autonomy is to enable quick response to dynamic environment and situations, including fault conditions that a spacecraft may encounter. The reliability of such autonomous capabilities depends on the quality of their knowledge of a spacecraft's health state. Model-based approaches to fault management, i.e. model-based fault diagnosis (MBFD), is one approach to continuously verify correct behavior in addition to diagnosing symptoms to estimate the spacecraft's health state.**

The proper functioning of MBFD is dependent on 1) the quality of the model that is analyzed and compared to the outputs of on-board sensors to estimate the system's health state, and 2) the correct functioning of the diagnosis engine that interrogates the model and compares its analysis to observed system behavior. We are currently developing Verification and Validation (V&V) approaches to provide the necessary confidence that MBFD systems are correctly estimating the health of on-board spacecraft components and systems. Our work is intended to narrow the gap between the rapidly maturing field of Model-Based System Engineering (including MBFD) and the less-well understood area of identifying and applying appropriate V&V techniques to MBFD.

Our effort is investigating three areas: 1) developing V&V techniques for the diagnostic model, 2) developing V&V techniques for the diagnostic engine in isolation, and 3) developing V&V techniques for the diagnostic engine and model in combination. This paper describes the work we have completed in the first area. We describe our approach to selecting a system to be represented by a model, the approach to modeling the system, the verification approach we developed for the model, and the results of the verification activity. We conclude with a description of the work remaining in the last two areas, which will be addressed over the next two years.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. BACKGROUND 2
3. MODEL-BASED FAULT DIAGNOSIS 2
4. APPROACH 3
5. RESULTS 11
6. FUTURE WORK 11
ACKNOWLEDGEMENTS 12
REFERENCES 13
BIOGRAPHY 13

978-1-5386-2014-4/18/$31.00 ©2018 IEEE

1. INTRODUCTION

Current state-of-the-art at JPL and across the aerospace industry [1] is to use monitors and responses to detect symptoms of off-nominal conditions after faults occur, and to respond with pre-defined sequences (see Figure 1). However, these symptom-based approaches can be fooled by sensor failures and environmental variation, or by commands that may intentionally put the system into an off-nominal state. In fact, monitors and responses are often turned off during highly critical events due to potentially incorrect fault identification and response. This approach also cannot verify healthy behavior, but only detects pre-defined failures that are being monitored. It is also prone to design escapes if a fault has not been considered during design time.

Model-based fault diagnosis (MBFD), on the other hand, diagnoses based on a model of how hardware components are expected to function and compares multiple sources of information (e.g., sensors, commands and relationships to other components) to establish the health state of each component. However, these approaches have only been demonstrated twice on flight missions (i.e., DS1 [2] and EO-1 [3]): although the demonstrations were very promising, this approach has not been formally verified or validated. Another example is a model-based system very similar to the one chosen for this work which was integrated with actual spacecraft FSW and ported to an engineering model of a late 1990's era flight computer. In hardware-in-the-loop (HWIL) tests this system was able to diagnose simulated faults in real time with significant margin in computing resources [4]. Although this was another successful demonstration of a MBFD technique, it did not provide full verification of the system.

The effort described in this paper is to develop Verification and Validation (V&V) approaches for a MBFD system that ensures it is correctly estimating the health state of on-board hardware components. Our objectives are:

- Develop and demonstrate techniques for checking MBFD correctness and coverage/completeness. The result will be a systematic way to verify MBFD correctness which will provide greater confidence than current V&V practices do for monitor/response systems.

- Develop and demonstrate MBFD performance analysis techniques (e.g., false-positive and false-negative diagnosis rates). The result will be a systematic way of analyzing MBFD performance that will provide greater confidence than what current practice provides for monitor/response systems. The structure of the diagnostic models and the way in which they interact with the diagnostic engine will provide information to improve model performance that would be difficult for monitor/response systems to provide.

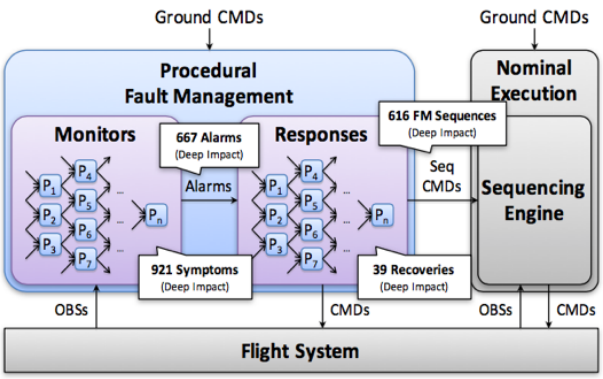


Figure 1 - Traditional monitor-response approach to Fault Management

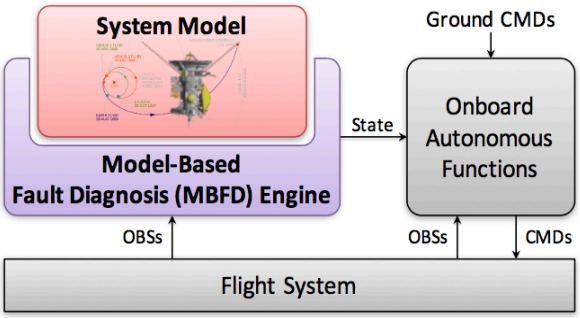


Figure 2 - Model-Based Fault Diagnosis Architecture

2. Background

Autonomy has been identified as a critical emerging area for JPL. One motivation for desiring onboard autonomy is to enable quick response to anomalies and dynamic environmental conditions and science opportunities. The reliability of such autonomous capabilities hinges on the quality of its knowledge of a spacecraft's state of health. However, current monitor-response Fault Management (FM) approaches (see Figure 1) do not diagnose the health state, but simply respond to a pre-identified set of symptoms from the sensor data. In situations not conforming to such well-defined cases, current FM approaches provide no information of system health, performance, or availability, and are thus inadequate for the needs of autonomy. In contrast, model-based approaches to FM, i.e. model-based fault diagnosis (MBFD), continuously verify correct hardware behavior in addition to diagnosing symptoms to estimate the health state. An onboard autonomy capability can then use health state estimates to decide how to react (see Figure 2). As such, MBFD is fundamental and foundational to enabling system-wide autonomy.

While MBFD has been developed through decades of research, the techniques for adequately verifying and validating MBFD are not well understood. Work by Pecheur et al. in 2002 [5], [6], [7] has surveyed various approaches to V&V MBFD, but there is still work to be done to enable a practical means to V&V MBFD. Other approaches have focused on simulations for verification [8] and developing tool chains [9] to partially automate as well as contain or bound the test domain. These approaches encompass system-level validation while other techniques such as parametric testing (using Monte Carlo methods for sensitivity analyses) target model validation [10]. As explained in [11], there exists a gap between NASA software requirements and standards for models in the case where models are part of the flight software. This gap makes it difficult to link verification of MBFD to requirements and to consider model V&V as a separate activity from system-level V&V.

Hence, this effort will identify and develop verification and validation (V&V) techniques that are appropriate for MBFD technologies. Without V&V of MBFD, the estimated health state produced by MBFD may be unstable or uncertain, or MBFD may cause unacceptable delays or excessive use of on-board computing resources. Without proper V&V of MBFD, the decisions made by other autonomous systems based on the MBFD's estimated health state will be held in low confidence, often necessitating review by spacecraft operators, and thus diminishing their value. Thus, V&V of MBFD is the necessary first step in enabling other autonomy capabilities.

3. Model-Based Fault Diagnosis

Fault Diagnosis Overview

A number of promising FM diagnosis techniques have been developed over the past several decades to provide faster hardware fault diagnosis and more effective responses to fault conditions than the currently-used monitor-response paradigm. Many of these results have been reported at the International Workshop on Principles of Diagnosis (e.g., [12], [13]), which was first held in 1989. FM techniques such as case-based, rule-based, and model-based approaches were reviewed for relevance and appropriateness to meet embedded real-time constraints that are required for spacecraft, rovers and other spaceflight systems. Approaches such as case-based and rule-based techniques are based on *a priori* knowledge and characterization of past faults, or enumeration of symptoms of known failure modes, and therefore are limited by the designers' ability to capture the symptom/fault information. The task of checking the completeness/coverage of case-based and rule-based techniques is challenging in part due to the many-to-one relationship between a symptom and its possible causes, which is exacerbated by the number of hardware configurations.

Model-based techniques perform system monitoring, fault detection and identification by reasoning from first principles, based on an understanding, i.e. a model, of how the system is expected to perform. These techniques do not rely on pre-defined symptoms of failures, and thereby overcome the one-to-many relations between a symptom and its possible causes. Instead, models follow nominal execution, and highlight when a system no longer performs as expected. There are a number of model-based fault management (MBFM) techniques, each with their strengths and weaknesses. We have chosen an MBFD technique based on constraint suspension [14].

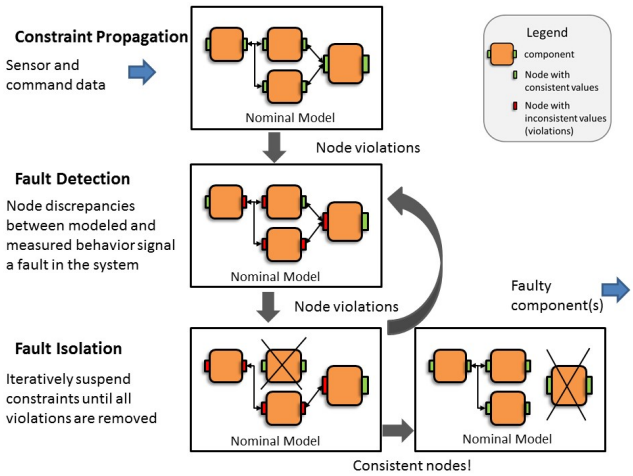


Figure 3 - Model-based fault detection and isolation checks system data consistency at component nodes and systematically determines which component caused the off-nominal condition.

MONSID Overview

The Model-based Off-Nominal State Isolation and Detection (MONSID) system is an implementation of constraint suspension extended to electro-mechanical systems [15], and is currently being developed by Okean Solutions under the Small Business Innovation Research (SBIR) program. It was first prototyped in the MATLAB/Simulink environment [16] and a C++ version intended for deployment. MONSID consists of a core inference engine that diagnoses faults from user-supplied models of the system given measurement and command data. As shown in Figure 3, measurements and command data are fed to the model representing the system. Models are made up of interconnected components that capture nominal behavior of system elements. In a model of a spacecraft, for example, components could represent various subsystem hardware such as sensors and actuators. These data are then propagated through the models via forward (input to output) and reverse (output to input) constraints as indicated by the arrows in the figure. Values resulting from the forward and reverse propagation are compared at component boundaries (nodes). Faults are detected by the inference engine when data from sensors, commands, and derived from models are no longer consistent at a node (typically several nodes). At that point, fault isolation begins via constraint suspension, i.e. systematically suspending data propagation across model components until data at input and output nodes are once again consistent. A fault is isolated to a component (or set of components) when that component's suspension results in consistent data throughout the model. In other words, if a component suspected of having a fault is removed from the model (its constraints are not processed) and no node violations occur on the remaining components then it must have been faulty. Because node discrepancies imply deviations from nominal behavior predicted by the models, this method can potentially detect anomalous and degraded states in addition to failed states.

4. APPROACH

In this effort, we investigate the three major facets of V&V for MBFD: correctness, completeness and performance. As shown in Figure 2, MBFD is composed of two parts, a system model that is specific to the system to be diagnosed and a generic diagnosis engine that is independent of the system to be diagnosed. Due to this key characteristic of MBFD, it is sufficient to check the correctness and completeness of the system model independently from the correctness and the completeness of the diagnosis engine. For performance, however, a system model and a diagnosis engine cannot be checked independently since the performance of a MBFD is a function of the diagnosis engine properties, the model size, and the model complexity. Over the course of this effort, we are developing approaches to checking the correctness and completeness separately for system models and diagnosis engines and developing approaches to checking the performance of MBFD as a whole. We are demonstrating the newly developed techniques on MONSID [16], which, in addition to this work, is also being used on JPL's research and development of autonomy capabilities.

System Model Overview

This year's effort focused on identifying techniques to determine the correctness of system models to be queried by the diagnostic engine. Our goal was to demonstrate the feasibility of determining model correctness by adapting acceptance and integration tests procedures and results used to evaluate flight and engineering units to the verification of the models. Reviews of current missions and discussions with engineers for those missions led us to focus on the Soil Moisture Active Passive (SMAP) mission [17], an Earth science mission designed to measure and map Earth's soil moisture and freeze/thaw state to better understand terrestrial water, carbon and energy cycles. Based on the discussions with the SMAP Fault Protection engineer, we have determined that fault protection issues related to Guidance, Navigation, and Control (GNC) would be a good candidate for the following reasons:

a. All missions encounter important GNC issues. In particular, the fielded behavior of GNC hardware may differ from that expected during design, especially when the hardware is operating in off-nominal conditions.

b. Unlike other areas, GNC systems and components tend to have similar behavior from mission to mission, although components (e.g., Inertial Reference Units (IRUs) or Stellar Reference Units(SRUs)) might not behave identically. However, they are often similar enough that adapting models from one mission to another is expected to be straightforward. By contrast, areas such as avionics tend to have much more mission-specific behavior.

and the development teams had moved to new tasks supporting other missions, resulting in significant effort to search for archived information. Since the morphology and level of detail of the models we developed are similar to those developed for SMAP and other missions, we are confident that future missions developing or adapting diagnostic models will need significantly less effort than we required, since domain experts will be developing the information required to construct the models at the same time the models are being developed.

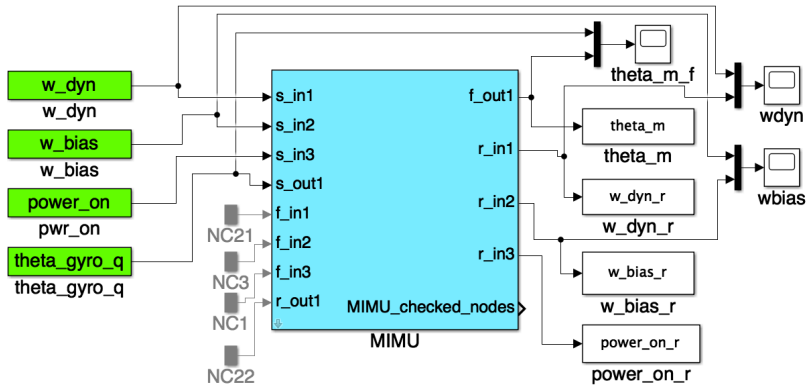


Figure 4 - MONSID Model of IRU

c. Although each mission has specific GNC challenges (e.g., a spinning antenna for SMAP), there are many commonalities that can be modeled. For example, nearly every mission needs to deal with situations in which different sensors provide different information. For example, the IRU may be indicating that the attitude is within expected limits, while the SRU may indicate otherwise.

We decided to develop a simplified model of the SMAP mission GNC subsystem consisting of four types of hardware components – an Inertial Reference Unit (IRU), the Reaction Wheel Assembly (RWA) and associated tachometers (TACH), and the Stellar Reference Unit (SRU). Despite its simplicity, the model provides a useful example from a real mission having real fault protection issues identified with it. Figure 4 shows one of the components we constructed, in this case the IRU. This component was implemented in the Simulink version of MONSID.

Developing System Models

The models of the SMAP GNC components and the simplified GNC subsystem were developed using information from the SMAP mission library, including Functional Design Documents (FDDs), design documentation, and supplier specifications (all three GNC components were acquired from outside suppliers).

The most time-consuming aspect of the model-building task was identifying and retrieving the information required to construct the models. We attribute this to two factors. First, none of the team members have the same level of familiarity with GNC systems as the mission developers. Second, the models were developed after the mission itself was launched,

We identified two issues to keep in mind when developing diagnostic models. First, in addition to describing the relationship between the model's inputs and outputs ("forward constraints") the diagnostic models also need to describe relationships between the outputs and the inputs ("reverse constraints"). The diagnostic engine querying the model performs a sequential relaxation of these constraints when it detects model output that is inconsistent with the observed system state until it can identify a set of causes of the inconsistency. Our experience indicates that up until now, system models at JPL have been developed to include only the forward constraints – there may be some effort required by system engineers to learn how to properly include reverse constraints.

Second, there is always the question of the level of abstraction at which a diagnostic model should be developed. There is no analytic solution to this issue, but some of our experience may be helpful. Models that are too coarse may not be able to detect or identify certain classes of anomalous behavior, such as slow degradation or progressive resource exhaustion. Models implementing a high-fidelity representation of a system component will allow more confident and detailed diagnosis of fault conditions than more abstract models, but can quickly become too complex to query within the required time. Also, model uncertainties may necessitate setting very broad fault detection thresholds to avoid false positives at the cost of mis-detecting actual faults. The principles for producing a good model discussed in [18] can be helpful, particularly the short discussion on the "smallest sufficient model." As a practical matter, diagnostic models may be implemented at varying levels of abstraction, with those portions of the model about which there is the

greatest concern about failure-related mission risk being specified at higher levels of detail.

As part of the verification process, it will be necessary to demonstrate that the model is able to detect and identify off-nominal as well as normal operation, meaning that simulators and analysis tools used to verify the model must be capable of exhibiting off-nominal behaviors. It is extremely unlikely that there will be sufficient resources available to examine the model's behavior under all possible off-nominal conditions, so it will be necessary to identify those fault conditions which analysis or operational and test data from earlier missions using the same or similar components indicates pose the greatest risks to mission success. Those fault conditions may be found by analyzing anomaly reports from previous similar missions, identifying the most commonly-occurring fault conditions or those with the greatest effect, then constructing a model that will display those behaviors given the appropriate inputs. Fault tree or failure mode effects analyses may be consulted in addition to anomaly reports to further refine the set of highest risk fault conditions.

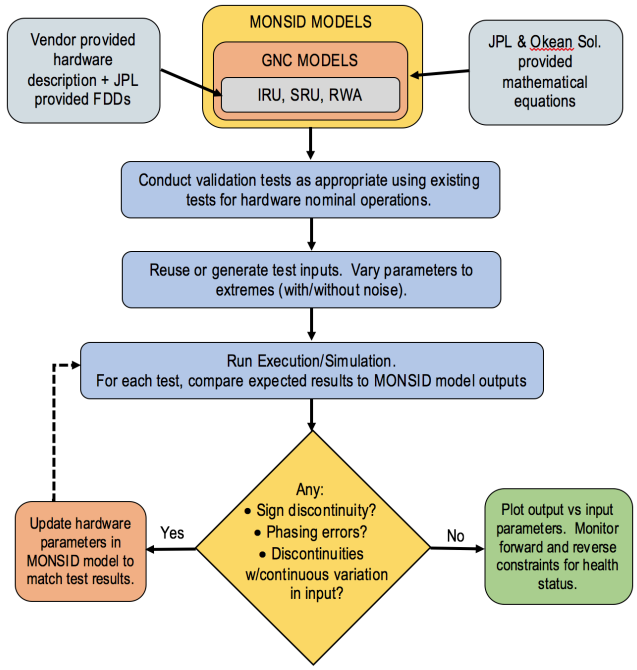


Figure 5 - Establishing a V&V Methodology

Developing A Model V&V Methodology

To ensure the correctness of our GNC Model, a robust Verification and Validation architecture is aimed to be established.

For our first year's work, we have adopted the traditional methodology to implement verification and validation of our models, as shown in Figure 5. In our case, the GNC components were designed and implemented based on their requirements. Then the implemented and integrated system and its components were verified using the corresponding verification methods that include procedures and tests. Similarly, we have developed a GNC system model based on the functional requirements of the system and its component, which will be verified using the corresponding verification methods.

The correctness of a system model can be checked against two different reference points: the functional requirements of the system and the deployed system itself. During the design phase of a mission, the model can be verified and validated against the functional requirements and during the operations phase of a mission, the model can be validated by checking it against the actual telemetry gathered during flight. Our current V&V approach allows us to check our model against the functional requirements as the reference point.

Our plan is to execute the following steps using a cohesive V&V approach as shown in Figure 5.

- Identify V&V Procedures: Hardware acceptance test conditions and test data will be used to perform V&V on the component and system models. For actuator models, the inputs to the actual hardware will be commandable and outputs will be measurable. For sensor models, however, the inputs (other than SW commands) will be used using motion and environment data such as rates, accelerations, etc. to create test cases.

- Identify MONSID simulation capability to perform V&V: A simulation/execution capability will be used to generate the types of inputs and outputs specified by the test cases. The simulation capabilities provided by MONSID can execute the model to simulate the functionality of the corresponding hardware, allowing us to use the model as if it were a hardware testbed. The model execution engine will run the forward and reverse constraints of MONSID models and pass data between components in accordance with the model topology.

- Identify Model constraints: The GNC component and system model parameters will be utilized for verifying MONSID model correctness. MONSID components are developed to model the expected behavior, under nominal operating conditions, of physical hardware. Because MONSID models translate behavior from outputs to inputs (reverse constraints), stimulants representing what are normally outputs of the real hardware are also needed as inputs to the MONSID models. Model outputs (forward constraints) may include analog and/or digital quantities depending on the type of HW component being modeled. Again, because of reverse constraints, model outputs may include commands and input power.

Model Testing

The procedures to test the MONSID models and compare their behavior to that of the SMAP flight units and engineering models were developed from test procedures and test results from both JPL and the component suppliers. We were able to identify and adapt component and system-level

tests developed by SMAP engineers to verify the functionality and behavior of the individual components and the integrated system for nominal operational scenarios.

a. Component Level V&V of Model Correctness:

The MONSID component level is the smallest encapsulation of hardware behavior that may correspond to a line-replaceable unit (LRU) of a spacecraft. Hardware is generally verified in the physical system using acceptance tests at component level. We have utilized existing hardware acceptance tests, or equivalent, to verify MONSID model components. In most cases, it was possible to use hardware test data directly by feeding into the simulation engine. In a few other cases, it was not possible to use hardware test data directly either because of the limited functionality of a component model or because the required data were not available or not in a usable format. Missing or limited functionality of any of the MONSID GNC components will be enhanced in the future. The existing functionality of each of the used GNC components was tested using the following validation tests to ensure component level model correctness:

i) IRU MONSID model validation test:

The IRU MONSID model is based on the model of the Honeywell manufactured Miniature Inertial Measurement Unit (MIMU) that processes its 3 axis Ring Laser gyro data at 200Hz to provide the inertial measurements of the spacecraft. This model outputs 3-axis spacecraft angular position as an accumulated angle over a specific duration of time. The model includes modeling coning compensation, noise, scale factor errors and two levels of quantization. It does not include generation of disturbance forces and torques from the internal dither mechanism.

To test the IRU MONSID model (Figure 4), we chose a MIMU phasing test procedure, developed at JPL, that had been run on flight IRU and for which the expected and actual results had been documented. Our adaptation of the test procedure monitored the following set of IRU forward and reverse constraints, which were processed by the IRU MONSID forward and reverse simulation scripts:

- Forward Constraints: Accumulated gyro angles
- Reverse Constraints: True inertial spacecraft rates, Gyro bias

The objective of testing the IRU hardware was to determine whether the IRU could sense known Earth rotation rates in different orientations. This goal was retained when testing the IRU MONSID model.

The true inertial spacecraft rates (Earth rotation rates in this case), gyro bias and noise parameters were provided as inputs to generate test data for nominal operation. The model output, processed as the change in sensed angle divided by the length of a time interval (angular rates along 3 axes), was compared to the expected values taken from the test procedure used for the flight unit. In this case, the actual results matched the expected results, confirming correct nominal behavior for the model. The comparison is shown graphically in Figure 6

The MONSID model phasing test results showed that the forward constraint values match MIMU measured accumulated angle as represented the Figure 7. The simulation script also processed the comparison of the reverse constraints with the test data inputs. Even though random noise was the most significant error source, the model reproduced similar characteristics. The plots shown in

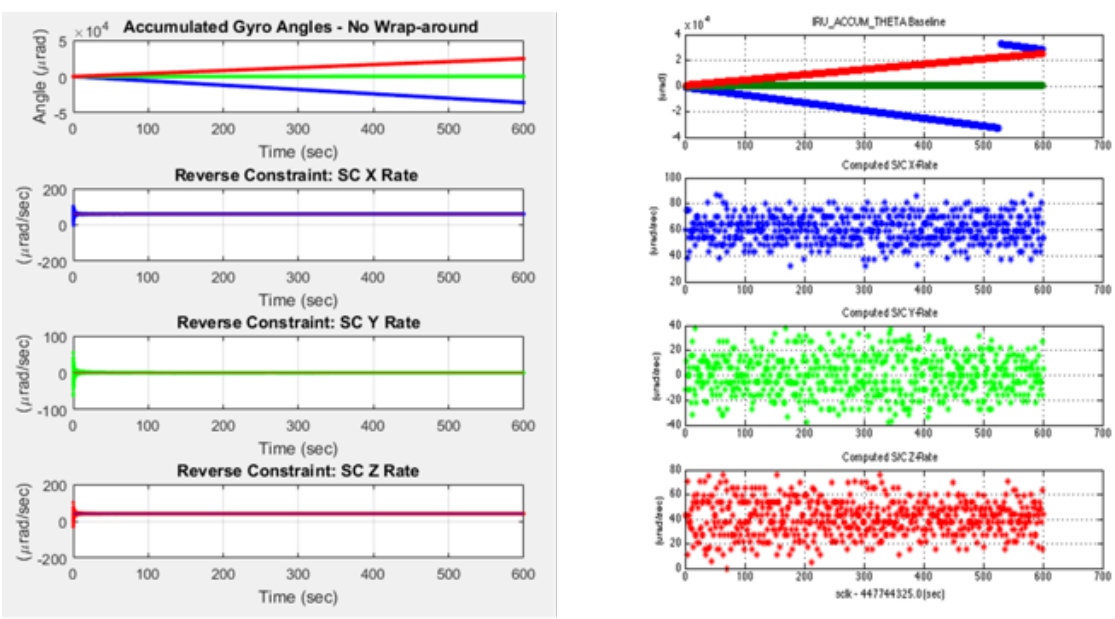


Figure 6 - left) MONSID model accumulated gyro angles and S/C rates match SMAP generated phasing test plots and right) true S/C rates

Figure 8 confirmed that the IRU MONSID model spacecraft rates (reverse constraints) converge on true spacecraft rates after 0.5 seconds.

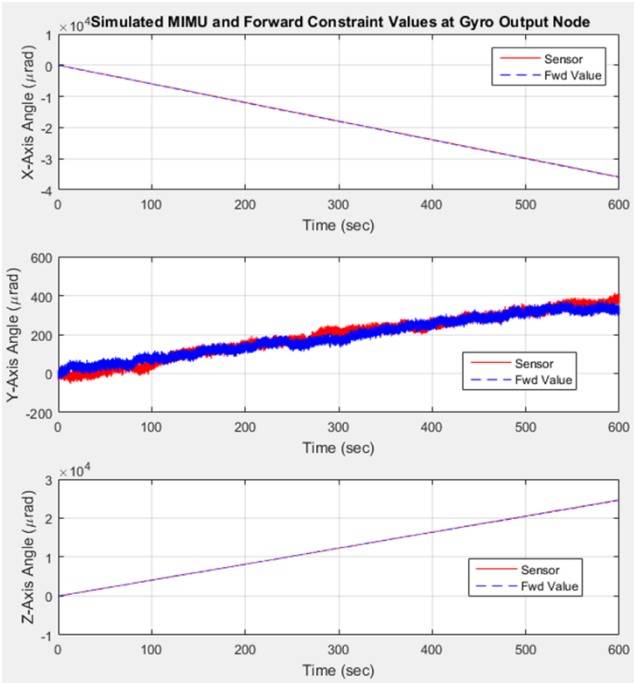


Figure 7 - MONSID Model Phasing Test Results – Forward values match simulated IRU accumulated angle

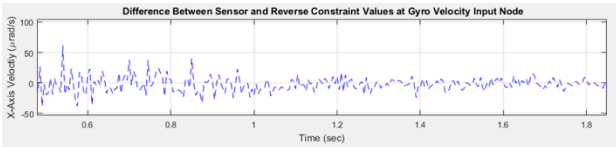


Figure 8 - MONSID Model Phasing Test Results – Plots show model converging on true signal < +/- 50 urad/s

ii) Reaction Wheel Assembly-Tachometer MONSID model validation test:

The RWA-Tach MONSID Model of a reaction wheel unit consists of two fundamental pieces at its core, the reaction wheel model and the tachometer model. The mathematical model of the RWA-Tach MONSID model has been extracted from the description of the Honeywell manufactured HR series Reaction Wheel Assembly. A torque command, encoded as a ±10V analog signal, of specific duration is provided as one of the inputs to the RWA-Tach MONSID model. Then, the model outputs the torque supplied to the reaction wheel and the wheel angles using the RWA block and also the corresponding tachometer pulse stream using the Tach block, at 8Hz. The tachometer count increments or decrements based on the value of the direction signal and will wrap to zero from full count when incremented or to full count from zero when decremented. Note that the System Level GNC Model integrates all four reaction wheel models to generate the net torque supplied to the spacecraft. The current RWA MONSID model includes viscous friction disturbance torque. Highly non-linear effects such as Dahl friction were not included in order to reduce model complexity. We may include these additional disturbances in future iterations if needed to improve model fidelity.

To test the RWA-Tach MONSID model shown in Figure 9, we chose a supplier provided hardware acceptance test and also an existing test procedure, developed at JPL, which had been run on a flight model of the RWA-Tachometer assembly. For both of the tests, the expected and actual results had been documented. Our adaptation of the test procedure monitored the following set of RWA/Tach forward and reverse constraints:

- Forward Constraints: Total wheel bearing torque, Reaction wheel angle, Incremental tachometer counts
- Reverse Constraints: Commanded torque

The objective of these two was simply to send torque commands to the RWA for a specified number of time intervals and then read the final tachometer count.

The final tachometer count, processed as the cumulative count of incremental tach counts from each command over all time intervals in the test, was then compared to the expected values taken from these test procedures. In this case, the actual results were close to the expected results, but did not match accurately as shown in Table 1.

Our analysis indicated that the differences in the actual vs. expected tach counts are mainly due to a mismatch between the torque parameter used in the model and the actual friction torque in the hardware. These results are being used to improve the accuracy of our models.

iii) SRU MONSID model validation test:

The SRU MONSID model shown in Figure 10 was developed to take the true spacecraft attitude quaternion as primary input, perturb it with some random noise and then output the perturbed quaternion data at 8Hz. It should be noted that the initial SRU model for this effort is intended to only perturb a truth quaternion with cross boresight and along boresight noise; none of the features like Star catalog file, Velocity aberration etc. will be included in the model. However, future versions of the model may include the additional features. The noise processes have been included in the model to accurately reflect the types of noise that can arise within the SRU. Values for each noise source were provided in documentation from the SRU supplier. Those sources are represented as independent random processes having a normal distribution with a mean value of zero.

To test the SRU MONSID model, we chose a SRU functional test procedure, developed at JPL, that had been run on flight SRU and for which the expected and actual results had been documented. Our adaptation of the test procedure monitored the following set of SRU forward and reverse constraints,

which were processed by the SRU MONSID forward and reverse simulation scripts:

- Forward Constraints: Measured Earth-Centered Inertial (ECI)-SRU attitude quaternion in SRU axes, measured ECI-SRU attitude rate in SRU axes

- Reverse Constraints: True inertial to spacecraft attitude quaternion

We analyzed this data thoroughly as there was a mismatch between the signs of the MONSID SRU and SMAP generated quaternions. Subsequent testing on the full model (Section 4 b) has resolved this issue as the SRU was able to sense correct quaternion values for that test. We concluded that the SMAP documentation for this test had a typographical error.

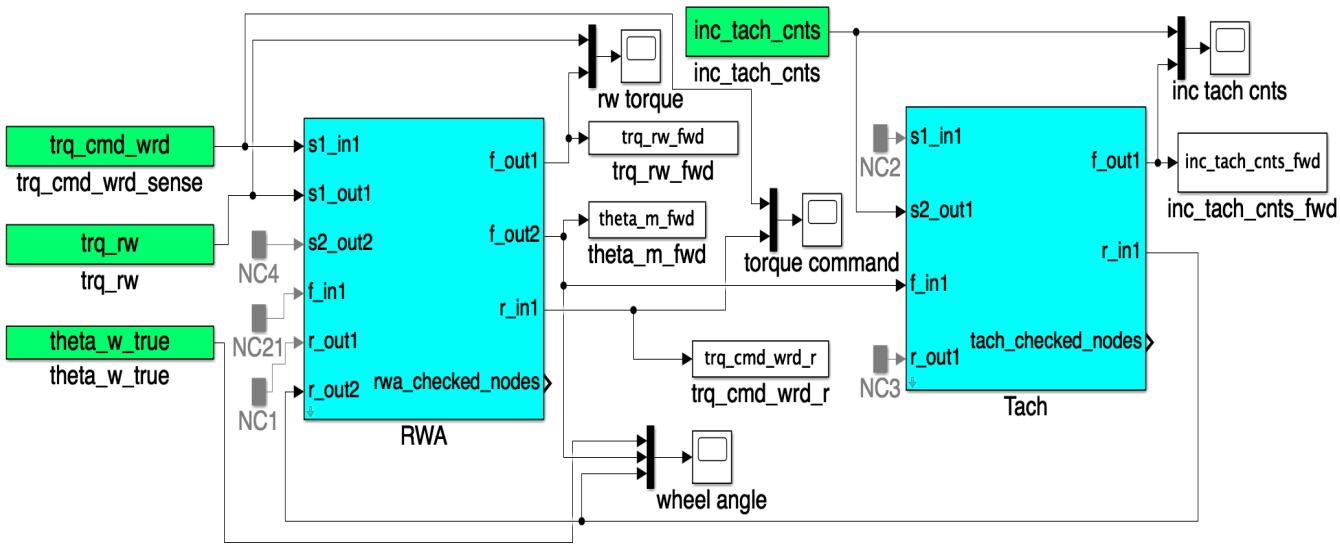


Figure 9 - RWA-TACH MONSID MODEL with Forward and Reverse Constraints

Test Case	Torque Command	Total Duration	Expected Wheel Speed	Expected Tach Count	MONSID Simulated Wheel Speed	MONSID Generated Tach Count
JPL provided flight hardware test: Test case 1	0.5 V(initial spin to wheel spin down)	25 sec	5 rpm	36 tach pulses	**5.8 rpm**	**23 tach pulses**
Supplier provided hardware acceptance test: high rate spin 1	10V(high rate spin) (0.254 nm measured)	622.4 sec	3900 rpm	1,139.36 tach pulses/sec	**3,794 rpm**	**1,138.2 tach pulses/sec**
Supplier provided hardware acceptance test: high rate spin 2	10V(high rate spin) (0.253 nm measured)	955.2 sec	6000 rpm	1,738.55 tach pulses/sec	**5,790 rpm**	**1,737.0 tach pulses /sec**

Table 1 - RWA-Tach MONSID Model validation test results for different test cases

The objective of this test was to determine if the SRU is able to supply correct SRU to spacecraft body frame quaternion output if the initial rate and attitude of the spacecraft and the mounting orientation of the SRU on the spacecraft are known. For a specific truth data, the output quaternion, perturbed with cross boresight and along boresight noise and processed over length of a time interval, produces the results shown in Figure 11 and Table 2.

True Rate (deg/sec)	True Initial Attitude Quaternion	SMAP Generated SRU To Body Frame Quaternion	MONSID Generated SRU To Body Frame Quaternion
(0,0,0)	[0,0,0,-1]	(-0.52200138, -0.02279096, 0.85182856, 0.037191)	**(-0.52199856, -0.022794080, 0.85183046, -0.037185743)**

Table 2 - SRU MONSID functional test results

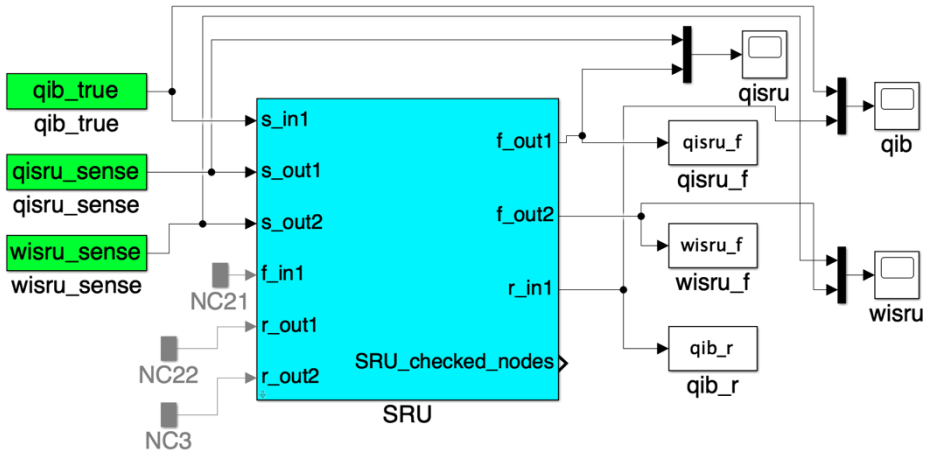


Figure 10 - SRU MONSID Model

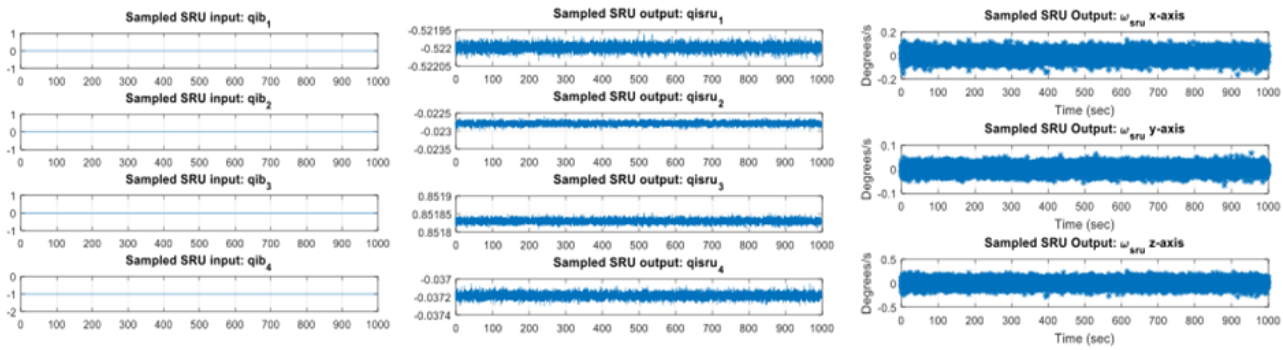


Figure 11 - SRU MONSID Model generated ECI-SRU Frame Quaternion and ECI-SRU attitude rates in SRU axes

b. System Level V&V of Model Correctness:

The MONSID system model level is the network of interconnected components that together describe a complete system or mode of operation, such as a reaction wheel based GNC. At this level, verification tests typically involve control system software. Although MONSID models at this time do not include models of the flight software, the commands generated by the flight software are inputs to MONSID.

The GNC MONSID model has been developed as the main diagnostic model for performing Assurance of Model-Based Fault Diagnosis (AMBFD) and is based on the SMAP GNC requirements, design documentation, specifications supplied by vendors of GNC components and V&V artifacts. The GNC model encompasses SMAP operational modes involving reaction wheels and fine attitude sensors. The SMAP spacecraft had a large spinning payload attached to the de-spun bus via a boom. In order to have a more typical example of a LEO spacecraft, the payload was not included in the GNC model. This simplified GNC model has a processing cycle of 125ms, incorporates the previously defined sensor and actuator components models (the IRU and SRU are the sensors and the RWA is the RWA), and also includes a Dynamics Kinematics Pseudocomponent (DKP). The purpose of the DKP is to provide linkage between the outputs of the actuator and the inputs of the sensor. This component implements a simple, linearized physics model of the spacecraft dynamic environment during mission operations. It is used to generate signals such as true spacecraft rates and accelerations that become inputs into models of sensor hardware. In MONSID models, these components are in between actuator components like reaction wheels and sensor components such as the SRU and IRU. The MONSID model of the simplified SMAP GNC, implemented in Simulink, can be seen in Figure 12. The blue boxes represent the 10 hardware component models. The DKP component is shown in yellow, to distinguish it from a true hardware component. The smaller green boxes represent command and sensor inputs to the model. Note that the connection lines between components include both the forward and reverse paths through the model. Figure 13 is a simplified representation of the GNC model where the four reaction wheel and tachometer components have been gathered into the RWA assembly; the reverse path and sensor inputs are not shown.

It is important to note that none of the planning and control elements of a classic GNC subsystem have been included in the model. We have abstracted those elements away, leaving only components essential to producing direct physical action of a spacecraft (i.e., RWA) or sensing the effects of those actions (the IRU and the SRU) as shown in Figure 13.

The initial body rates and attitude are provided as true inputs to initialize the GNC model. Torque commands (one command for each wheel of the RWA), are another set of inputs to the GNC model, that command the value and direction of the torque applied to each reaction wheel.

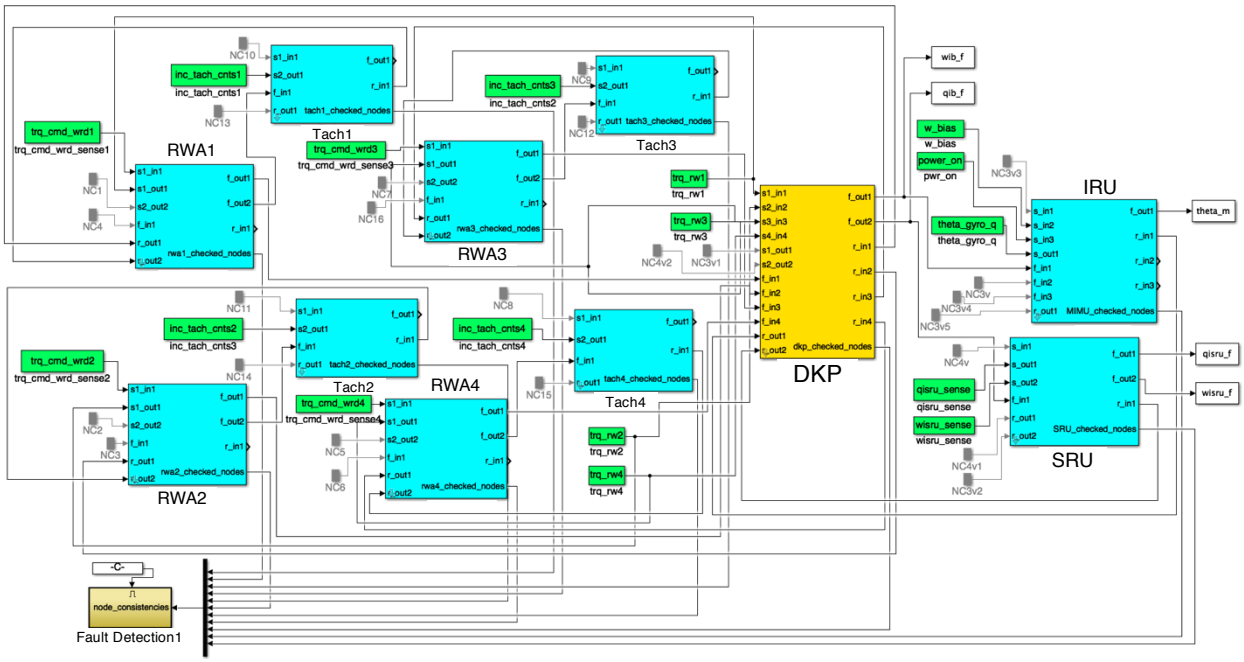


Figure 12 –MONSID GNC System Model

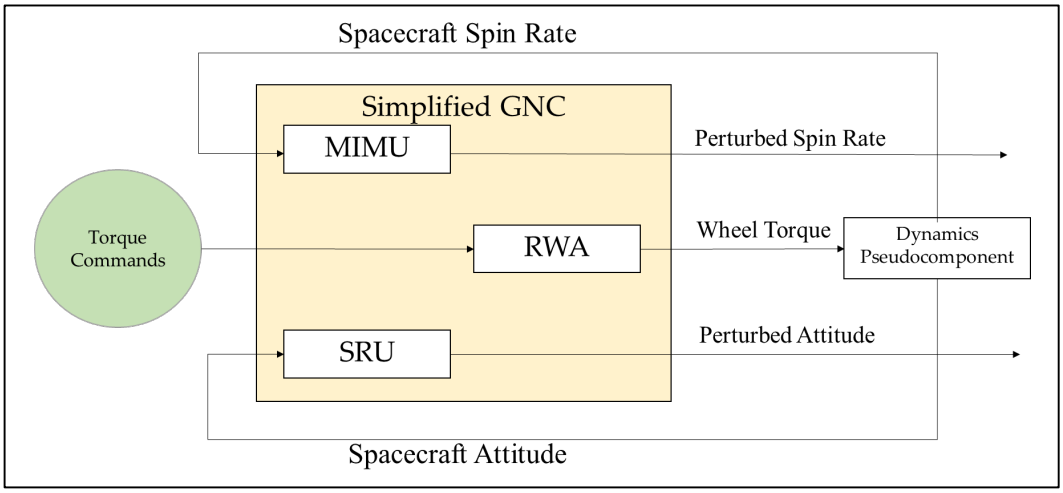


Figure 13 – Simplified MONSID GNC System Model

The GNC model outputs include outputs from the sensor models. The IRU output is the spin rate of the spacecraft which is the true body rate input mixed with noise generated by the gyro component of the IRU model. The SRU outputs include the perturbed attitude of the spacecraft which is the true attitude mixed with additive noise representing the root-sum-squared bias and random contributions. The second SRU output is the spacecraft attitude rate in the SRU frame. Additionally, the RWA model produces a reaction wheel torque and wheel angular rate in response to a torque command directed to that wheel by the user. The tachometer component generates incremental tachometer angles, which are outputs of the model. The DKP relates the reaction wheel torques to the spacecraft torque, and thereby the change in angular momentum, which in turn relates to the spacecraft spin rate along each axis, thereby coupling an actuator output (RWA torque) to a sensor input (IRU measurement of spin rate). We plan to perform additional V&V on the DKP component model as part of our future work.

We used the MONSID engine's simulation capability to run the GNC model with varioius test inputs and to monitor all the forward and reverse constraints of the GNC model, including the internally propagated constraints between the components. We performed end to end validation testing on the GNC MONSID model by sending torque commands of specified durations as inputs and then monitoring the corresponding IRU and SRU output values, during and after the application of torque impulses. We then compared these

output values along with the true inputs to the outputs of a JPL developed sensor based model verification test used on the SMAP program to validate GNC models in the GNC design and performance verification simulation.

We performed the end to end validation test by sending a ten second long 0.2 Nm torque command to one of the Reaction Wheels, while providing torque commands of 0 Nm to the remaining three wheels. We then monitored the MIMU and SRU measured values after sending the torque command. The initial body rate and attitude were defined to be 0 deg/sec and [0 0 0 1] respectively.

This end-to-end test verified that the model is consistent throughout. The values of the spacecraft body rates coming out of any component node were similar. The 'true' spacecraft rates coming out of the forward node of the DKP due to the applied torques were sensed accurately by the MIMU and SRU sensors under no noise conditions. These values also matched the expected body rates that were specified in the JPL provided verification test, as shown in Table 3. Additional tests to stimulate the remaining RWA components will also be run as part of the end-to-end testing.

Time Point (t)	Expected Body Rates	(A) "True" spacecraft body rates provided by DKP (forward values)	(B) MIMU measured body rates (forward values)	C) SRU measured body rates (forward values)
10 sec	[0, -0.00028, -0.00198] rad/sec	[-5.71E-06, -0.000277536, -0.001982401] rad/sec	[0, -0.00028, -0.00198] rad/sec	[0, -0.0003, -0.0020] rad/sec

Table 3 - GNC End to End Validation Results

5. RESULTS

Our results to date demonstrate the feasibility of adapting test procedures and results used to evaluate the functionality and behavior of flight mission components/systems for use in determining the correctness of diagnostic models developed from the specifications and design of the flight units. If diagnostic models are developed during the overall system development effort, our experience indicates that the effort involved in developing those models should not be much greater than effort required to develop models for other purposes (e.g., models developed by GNC designers to validate control algorithms).

In testing the models, we found that test procedures and expected results about a component's nominal functionality and behavior are more readily available than information about off-nominal behavior. For hardware components, some of this is because testing a component under off-nominal conditions could result in physical damage, and mission development efforts are not willing to take this risk with flight hardware or engineering models of that hardware. In this case, it may be necessary to develop additional verification procedures directly from a component's specification or design documentation. Techniques such as the version of combinatorial testing developed by Kuhn et al. [19] can be applied provided that 1) the requirements or design documentation describe off-nominal functionality and behavior in addition to nominal operations, and 2) the functionality and behavior described by those documents can be parameterized and those parameters grouped into equivalence classes.

Once the test procedures and expected results for off-nominal behavior have been developed, an infrastructure similar to that used for testing flight software design and performance would be used to verify the diagnostic models. Simulators allowing the injection of faults (e.g., SMAP's Control Analysis Simulation Testbed (CAST)) would be part of this infrastructure, and would used to examine the behavior of the diagnostic models under the off-nominal conditions specified in the test procedures. These would come into play in developing fault cases for verifying MONSID's fault detection and identification functionality. In these conditions, a correct nominal MONSID model would necessarily deviate from the simulation. This would allow verification of the models' correctness even under conditions in which it might not be possible to inject faults in the actual hardware components themselves.

6. FUTURE WORK

For the remainder of our work, we will identify techniques to assure the correctness and completeness of the diagnostic engine in isolation, and then do the same for the integrated diagnostic system and model. We will also identify appropriate techniques for assessing the performance of the diagnostic engine and the integrated diagnostic engine/model system. A short outline of our plans over the next two years is given below.

During the coming year, we will work in the following areas:

- Perform static analysis on the MONSID diagnosis engine and report on test results and coverage analysis.
- Identify applicable formal verification techniques for V&V-ing models and diagnosis engines.

These first two items fall into the category of demonstrating diagnosis engine correctness and completeness. For a diagnosis engine to be "correct" (i.e. for each input, it produces the expected output) the algorithm used by the diagnosis engine must be correct and its software implementation also must be correct. Similarly, for a diagnosis engine to be "complete" (i.e. it handles all possible inputs), the algorithm used by the diagnosis engine must be complete and its software implementation also must be complete. Checking flight software for correctness and completeness is a well-studied problem with many available techniques that can be readily leveraged and applied. Techniques range from architecture, design, and code reviews to various testing and static analysis methods and formal verification methods. We will survey existing

methods to identify approaches best suited for checking the correctness and completeness of MBFD engine, and mature them accordingly. To scope our effort, we will implement these techniques on a particular MBFD example technology, specifically MONSID. Note that V&V-ing the diagnosis engine is decoupled, thus independent from V&V-ing models. While the proposed approach will be demonstrated on MONSID, we expect the technique to be generally applicable to other types of MBFD diagnosis engines.

- Analyze and report on MONSID + Model memory foot print and runtime performance. It is important to determine these values to determine how MONSID may be used most effectively on flight systems for which computing resources are usually more constrained than those that are ground-based. In general, the required memory foot print and processing time is a function of the size and complexity of a model and the diagnosis engine characteristics. However, because the engine is fixed across applications, variances in computer resource usage will be due to differences in model complexity.

- Develop and demonstrate model completeness checking techniques. Quantitative knowledge of the extent to which a diagnostic model is complete with respect to the system specifications will increase the confidence of the system health estimates provided by MBFD. We will consider a system model's completeness to be defined by how much of a system has been modeled. Checking the completeness against the functional design is rather straightforward. We can compute the system topology based on the functional design and compare that against the topology of the system model. With topology defined as the list of components, their interfaces and their connotations, we are able to generate a topology from a model and also generate a topology from the functional design document. These two topologies can then be automatically checked for differences. While the proposed approach will be demonstrated on MONSID models, we expect the technique to be applicable to other types of MBFD models.

- Develop and demonstrate diagnosis resolution analysis techniques. The other important performance metrics for fault diagnosis are the diagnostic resolution (e.g. ability to diagnose a fault to one component vs. to a group of components), and the rate of false-positive and false-negative diagnoses. The diagnostic resolution is a function of a system's component and sensor topology. While analyzing the performance metrics such as memory foot print and processing time is straightforward, analyzing diagnostic resolution is not trivial because it is highly dependent on the design and implementation of the system to be diagnosed. We will investigate and develop the techniques for analyzing diagnostic resolution of MBFD will demonstrate the techniques using MONSID.

During the year after the goals above have been achieved, we will focus on the following areas:

- Perform false-positive and false-negative diagnosis rate analysis. The rate of false-positive and false-negative diagnoses is a function of system complexity and the level of model fidelity. As with diagnostic resolution, analyzing the rate of false-positive and false-negative diagnoses is not trivial because it is also highly dependent on the design and implementation of the system to be diagnosed. We will investigate and develop the techniques for analyzing this aspect of MBFD performance and demonstrate the techniques using MONSID.

- Develop and demonstrate formal verification techniques for model correctness checking. We will generate model checking techniques such as SPIN ([18], [20]) and the Symbolic Analysis Laboratory (SAL) tool suite [21]. We will investigate techniques for verifying the diagnostic engine in isolation and in combination with a diagnostic model. A key element in applying these techniques is developing correctness properties that must be satisfied: one set for MONSID alone, and a second set for the MONSID+diagnostic model combination. Based on the way in which constraint suspension operates, we think it likely that these properties will involve:

    - Guarantees that constraints are suspended in the specified order until an inconsistency is found.

    - If no inconsistency is found, a guarantee that all constraints have in fact been suspended.

    - Guarantees that constraints are only suspended a single time, and that there are no lack-of-progress cycles in the suspension of constraints (i.e., constraint A is suspended, then constraint B,…, then constraint A).

Although not directly related to the V&V of MBFD, additional future work for this program could include refinements to MONSID to handle situations in which a faulty component could not be unambiguously identified. For example, the removal of one of a component's constraints, while removing inconsistencies, could result in insufficient data to check other constraints in neighboring components. Ambiguity groups, discussed in [22], could be used to address this issue as part of future work.

ACKNOWLEDGEMENTS

Part of the research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through the internal Research and Technology Development program. The development of MONSID was funded under a Small Business Innovative Research (SBIR) Phase II contract with JPL.

REFERENCES

[1] L. Fesq, K. Fretz and M. Newhouse, "Report on the 2012 NASA Spacecraft Fault Management Workshop," 2012.

[2] Bernard D.E., Dorais G.A., Fry C., Gamble E.B., Kanefsky B., Kurien J., Millar W., Muscettola N., Nayak P., Pell B., Rajan K., Rouquette N., Smith B., Williams B.C., "Design of the Remote Agent Experiment for Spacecraft Autonomy," in IEEE Aerospace Conference Proceedings, Snowmass, CO, Mar. 21-28, 1998.

[3] Hayden, S.C., Sweet, A., Shulman, S., "Lessons Learned in the Livingstone 2 on Earth Observing One Flight Experiment," presented at the AIAA Infotech Conference, 26 - 29 September 2005, Arlington, Virginia, paper # AIAA-2005-7000.

[4] Kolcio K., Hanson M. L., Fesq L. M., Forrest D. J., "Integrating Autonomous Fault Management with Conventional Flight Software: A Case Study", IEEE Aerospace Conference Proceedings, Snowmass, CO, Mar. 1999.

[5] C. Pecheur and S. Nelson, "Survey of NASA V&V Processes/Methods," Moffett Field, CA, 2002.

[6] C. Pecheur and S. Nelson, "V&V of Advanced Systems at NASA," Moffett Field, CA, 2002.

[7] C. Pecheur and S. Nelson, "New V&V Tools for Diagnostic Modeling Environment (DME)," Moffett Field, CA, 2002.

[8] Lindsey, A.E., Pecheur, C., "Simulation-Based Verification of Autonomous Controllers via Livingstone PathFinder," TACAS 2004, LNCS 2988, pp. 357–371, 2004.

[9] Schumann, J. *et al*., "A Tool Chain for the V&V of NASA Cryogenic Fuel Loading Health Management," Annual Conference of the Prognostics and Health Management Society, 2014.

[10] Reed, E., Schumann, J., Mengshoel, O.J., "Verification and Validation of System Health Management Models using Parametric Testing," Infotech@Aerospace St. Louis, MO, 29-31 March 2011

[11] Kodali, A., Robinson, P., "Towards Accreditation of Diagnostic Models for Improved Performance," Annual Conference of the Prognostics and Health Management Society, Fort Worth, TX, Sept 29-Oct 4 2014.

[12] 28th International Workshop On Principles Of Diagnosis (DX'17) Web site: https://www.fbk.eu/-en/event/28th-international-workshop-principles-diagnosis-dx17/

[13] Feldman, A., Kalech, M., Provan, G., (editors), *Proceedings of the 24th International Workshop on Principles of Diagnosis*, June 2013.

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

[15] Fesq, L, "Marple: An Autonomous Diagnostician for Isolating System Hardware Failures," Ph.D. Dissertation, UCLA, 1993.

[16] K. Kolcio and L. Fesq, "Model-based Off-nominal State Isolation and Detection System for Autonomous Fault Management," in 2015 IEEE Aerospace Conference, Big Sky, MT, 2016.

[17] "Soil Moisture Active Passive", SMAP Web site: https://www.jpl.nasa.gov/missions/soil-moisture-active-passive-smap/

[18] Gerard Holzmann, "The Spin Model Checker: Primer And Reference Manual," Addison-Wesley Professional, 2003, pp. 101-126.

[19] Kuhn, D. Richard, Raghu N. Kacker, and Yu Lei. NIST SP 800-142. "Practical Combinatorial Testing." (2010). Available at http://nvlpubs.nist.gov/nistpubs/Legacy/-SP/nistspecialpublication800-142.pdf (Oct 2017).

[20] "Verifying Multi-threaded Software with Spin", SPIN Web site: http://spinroot.com/spin/whatispin.html.

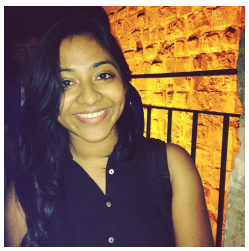
[21] "Symbolic Analysis Laboratory", SAL Web site: http://sal.csl.sri.com.

[22] Kolcio, K., Fesq, L., & Mackey, R., "Model-Based Approach To Rover Health Assessment For Increased Productivity," in 2017 IEEE Aerospace Conference, Big Sky, MT, 2017

BIOGRAPHY

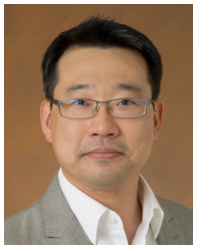
Allen Nikora *is a Principal Software Assurance Engineer in JPL's Quality Assurance Office, and has been with the Jet Propulsion Laboratory, California Institute of Technology for nearly 40 years. With support from NASA's Office of Safety and Mission Assurance, the U. S. Air Force Operational Test and Evaluation Center, the U. S. Naval Air Systems Command, and JPL internal funding, he has conducted extensive research in various aspects of software reliability engineering and defect analysis, and has developed tools to assess software reliability during test and operations. He has published extensively on his work, and has been active in the IEEE standards community, participating as a working group member and chair for a number of software reliability-related IEEE standards including all versions of IEEE Std 1633 and IEEE Std 982.1-2005. He received the*

NASA Exceptional Achievement award for contributing to the December, 1995 GALILEO entry into Jupiter orbit and relay of probe data, and the NASA Inventions and Contributions Award and Space Act Award for developing the CASRE software reliability tool. He holds a B.S. in Engineering and Applied Science from the California Institute of Technology, and a Ph.D. in Computer Science from the University of Southern California. He belongs to the IEEE, the IEEE Computer Society, and the IEEE Reliability Society.



***Priyanka Srivastava** is a System Verification & Validation (V&V) Engineer of the Flight Systems Engineering, Integration and Test section at JPL. She has over a 2 years' experience of working with Model-Based Systems Engineering (MBSE) applications at the Jet Propulsion Laboratory, California Institute of Technology. She is currently a V&V engineer for Orbiting Carbon Observatory -3 mission, where she is performing wide range of flight system testing and requirement management responsibilities. Priyanka has also tested and analyzed the internal memory organization and bus structure of the Descent Motor Control Assembly (DMCA) of the Mars 2020 flight system and identified major integration behaviors of the motor assembly with the rest of the flight system. She has also worked on modelling launch systems interfaces with flight systems using MBSE techniques, specifically for Europa clipper, and also helped in formulating the launch system requirements for those interfaces. Before starting full time at JPL, she has also worked as an intern at NASA Glenn where she performed MBSE tasks for modelling Near Earth Network for the Space Communication and Navigation(SCaN) group. Priyanka holds a Master's degree in Space Systems Engineering from University of Michigan, Ann Arbor.*

***Lorraine Fesq** is a Principal Engineer in the Engineering Development Office at the Jet Propulsion Laboratory, California Institute of Technology. She has over 30 years of aerospace experience that spans industry, government and academia, has worked all mission phases of spacecraft development. She has received a NASA Public Service Medal for her work on the Chandra X-ray Observatory, and a NASA Exceptional Achievement Medal for advancing the Fault Management discipline within NASA. Lorraine taught in the Aeronautics/Astronautics department at MIT while researching model-based diagnostic techniques. She organized both of NASA's Fault Management Workshops, which brought together Fault Management practitioners and experts from NASA, DoD, industry, and academia to share insights and to expose and address systemic challenges. Lorraine led a NASA-wide assessment and advisory team to review the Constellation Program and to recommend improvements to the program's Fault Management plans, designs, and organizational structure. Lorraine is the Lead for NASA's FM Community of Practice, is the co-Lead for NASA's Software Architecture Review Board, and serves as a Deputy Software Technical Discipline Lead with the NASA Engineering & Safety Center. Lorraine received her B.A. in Mathematics from Rutgers University and her M.S. and Ph.D. in Computer Science from the University of California, Los Angeles.*



Dr. Seung Chung is the Assistant Section Manager of the Planning and Execution Systems section at the Jet Propulsion Laboratory, California Institute of Technology. He has over 18 years of work experience in systems engineering and Artificial Intelligence, with expertise in model-based system design and analysis, planning and execution, and fault management. He received his PhD in Autonomy and MS in Aeronautics and Astronautics from the Massachusetts Institute of Technology.

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