Scaling Up Model-Based Diagnostic and Fault Effects Reasoning for Spacecraft

Gordon B. Aaseng[1]
NASA Ames Research Center, Houston, TX 77058

Eric Barszcz[2]
NASA Ames Research Center, Moffett Field, CA 94035

Henry Valdez[3]
Independent Design Analyses, Inc., Houston, TX 77058

and

Haifa Moses[4]
NASA Johnson Space Center, Houston, TX 77058

NASA developed and tested an automated model-based diagnostic and fault effects system and monitored the Orion Exploration Flight Test 1 (EFT-1) data from prelaunch to post-landing. One primary objective was the maturation of fault management technologies that will be required for autonomous operations in missions operating beyond Earth orbit by using a large scale model with actual mission designs and flight data, across a range of environments and system configurations in realistic operational conditions. A second primary objective was to explore and demonstrate methods for cost-effective model development and maintenance, including automated or semi-automated model development and use of models for multiple purposes from design analysis through operations. Both objectives included identification of remaining issues that will need to be resolved in order for programs to successfully adapt and deploy the technology in human-rated space systems. This paper describes the methods used to develop and test the system fault model, techniques to monitor the data and diagnose faults during flight operations, and the key findings based on the results of the demonstration.

Nomenclature

ACAWS = Advanced Caution and Warning System
ADIO = Analog-to-digital input/output
ARC = Ames Research Center
ATOM = Advanced Tool of Math
CAD = Computer Aided Design
EFT-1 = Exploration Flight Test 1
EM-2 = Exploration Mission 2
EPS = Electrical Power System
FDIR = Fault Detection, Isolation and Recovery
FMEA = Failure Modes and Effects Analysis

---

[1] Computer Scientist, Intelligent Systems Division, MS 269-1.
[2] Computer Scientist, Intelligent Systems Division, MS 269-1.
[3] President, IDA Inc, 16821 Buccaneer, Ste. 206, Houston, TX.
[4] Human Factors Engineer, Human Systems Engineering & Development Division, SF3.

GUI = Graphical User Interface
Hz = Hertz (cycles per second)
ISP = Information Sharing Protocol
MCC = Mission Control Center
PDU = Power Distribution Unit
RDS = Remote Diagnostic Server
RPC = Remote Power Controller
SysML = System Modeling Language
TEAMS = Testability Engineering and Maintenance System

I. Introduction

AUTOMATED diagnosis and determination of the effects of faults on system performance and reliability have been recognized as fundamental building blocks of highly efficient, automated and autonomous space operations[1]. Numerous technology research, tool development and demonstrations have been conducted to develop automated system level diagnostics, system health assessment, and automated decision-making, including a prototype precursor to the work described in this paper with the Ares 1-X test flight[2] and launch operations at Kennedy Space Center[3]. However, automated fault management systems have not been widely deployed in spacecraft, beyond localized sub-system Fault Detection, Isolation and Recovery (FDIR) and automated transition to a safe mode from which human operators make recovery decisions. Although the technology foundations have been proven, there are several impediments to Human Space Flight program organizations deploying fault management automation to meet their system requirements. These impediments include lack of demonstrated success in full-scale systems, lack of trust of automated decisions by human operators, and uncertainty about the cost of developing, testing and maintaining automated operations systems.

It is recognized that deep space human missions with light times on the order of several minutes will require much more dependence on autonomous operations and hence, automated decision tools that support highly autonomous operations with all immediate or near-term decisions made on board, either fully automatically or by the crew working only with the on-board systems[4]. To address these impediments to deployment, NASA Ames Research Center (ARC) developed the Advanced Caution and Warning System (ACAWS) and conducted system-scale model-based reasoning for automated failure diagnosis and effects of faults with the Orion Multi-Purpose Crew Vehicle's first test flight, Exploration Flight Test 1 (EFT-1) in December 2014. Monitoring was conducted from pre-launch through post-splashdown, using data transmitted to the ground, to determine the cause of failures and identify components that are affected by failures. Failure effects determination includes both loss of function and loss of redundancy of critical equipment, and coupled with fault diagnosis, provides a comprehensive view of the health of the spacecraft using a novel control center display user interface. The ACAWS project also demonstrated capabilities to assist operators using "what-if" queries that identify next worst failures to help operations teams to be prepared for the most critical system failures using the same model and technology. Post-mission analysis provided significant information to improve the focus of future technology development to continue moving toward overcoming the impediments to deployment of advanced health management reasoners and decision tools.

This paper describes how we addressed the needs for maturation of health management autonomy with the Orion EFT-1 flight, and used the flight analysis to identify several key remaining challenges needed to progress from technology demonstration to full-scale deployment. We begin with describing the purpose, goals and objectives of the project. A discussion of some of the challenges, both technical and programmatic, of developing system models and methods that were used to build models for this project will show how the methods can be applied to programmatic use of the technology on flight programs. An assessment of the usefulness and value of fault models for fault management systems engineering for the Orion Program is described next. Key features of the operational architecture used for diagnostics with the flight data will be described followed by a discussion of the system test and evaluation performed prior to the Orion EFT-1 flight. Operational execution, observations and findings from the Orion EFT-1 flight and post-flight analysis are described, followed by closing with a discussion of the advances in maturation, scalability and robustness that were achieved with the Orion EFT-1 flight and remaining challenges to be overcome before full-scale deployment will become a reality.

A key aspect of affordability of large-scale automated health management systems is the use of fault models for multiple purposes from early design through operations. The ACAWS project initially developed a system fault model from primary design documents with semi-automated, or "power-assisted" modeling methods, and demonstrated use of the model in design analysis activities, including channelization analysis to determine if wiring layouts resulted in

hidden cases of a single fault causing loss of critical function. Follow-on modeling added information required for diagnosis and effects determination, successfully demonstrating a multi-use model at a scale sufficient for current program needs.

The project focused on quality of health state information for correctness and usefulness in decision-making. Building trust in automated information systems requires that the information is correct under all conditions, configurations and environments, is at a level of detail useful for making response decisions and is readily understood by operators whether they are making decisions with the information or are monitoring the decisions made by automated reasoners.

Test and verification of the system health state information using a fault model presented some challenges. The correctness of diagnosis and effects information was tested and analyzed using a series of failure cases developed in coordination with Orion Program test systems. The Orion program conducted extensive testing with nominal data from test facilities and the vehicle itself flowed to the Mission Control Center (MCC) for vehicle and mission control systems integration and test. Nominal data was recorded from these full mission tests, and failure scenarios were developed by over-writing fault signatures onto the nominal data to verify diagnostics in a variety of mission phases. Even though not a full scale verification, the exercise provided substantial insight into the challenges and risks involved with test and verification of a large scale diagnostic and effects model.

To assure that the system produces the information most valuable for flight operations, a controlled evaluation with mission operations experts was conducted. Multiple failure scenarios were developed, varying in complexity of diagnosis and operational decisions, workload levels and difficulty. Evaluation runs were conducted in realistic control team settings to determine the effect that the system level health state information has on decision-making and situational awareness, and to identify the information attributes that maximize benefits. Both quantitative performance data and subjective feedback were collected and analyzed to determine the information and presentation features most useful to flight controllers, and where research and development is needed to meet operational needs for higher quality of health state information. Although the evaluation was conducted in a ground operations setting, the information can be extrapolated to on-board crew display needs as well.

This paper will explain the project goals and objectives, the ACAWS architecture and model approach, the associated development challenges, and ACAWS use in Fault Management Systems Engineering. It also presents the results and findings of the System Testing and Evaluations performed, and a real time direct application during the Orion EFT-1 mission. The paper provides conclusions, impediments to operational deployment and lays out recommendations for continued scalability, accuracy, correctness, usefulness of information and life cycle costs of health management system deployment.

II. Project Goals and Objectives

The EFT-1 ACAWS project was conducted to achieve two primary objectives:

- Demonstrate that system fault models can reduce the effort and improve the results of fault management analysis during the systems engineering and design phases of the project.
- Demonstrate that the same model can be used as the basis for operational fault diagnosis and determination of the functional and loss of redundancy effects of system faults that will be a central element of system autonomy requirements in deep space human exploration.

In order to address these objectives realistically and credibly it was important to scale the models to a significant portion of the spacecraft and maintain accuracy and correctness. The project was not to be a proof of concept in which shortcuts and simplifications would be used to work around difficulties. The project team collaborated with the Orion Program to select the Electrical Power System (EPS) as the primary domain of the model. The EPS was selected because it is a major critical system with interfaces to nearly all other systems, and connectivity data was readily available. The model included most of the electrically powered equipment in the spacecraft from major equipment such as flight computers and pumps, to the operations sensors needed in system controls. The model omitted the Development Flight Instrumentation to focus on the spacecraft elements destined to be permanent parts of the Orion design.

The project selected Qualtech Systems, Inc.'s TEAMS Designer ™ (Testability Engineering and Maintenance System) [5] as the fault management modeling tool, and its companion run-time reasoner TEAMS-RDS ™ (Remote Diagnostic Server) as the diagnostic engine. The project established some sub-goals to assess and demonstrate some key model development attributes, including:

- Model scalability. Build a model that includes a significant portion of the operationally significant faults, in order to show that model-based fault management can be accomplished within realistic development and operational resources.
- Accuracy and realism. Develop a model that remained true to the Orion systems, including correct and accurate failure modes as defined by the Orion Program Failure Modes and Effects Analysis (FMEA) without simplifications or workarounds.
- Model Development Efficiency. Be able to assess and demonstrate modeling methods that could be done with moderate costs and that could remain accurate as the design continued to change and mature. Automated import of available data was a key aspect of achieving the objective.

III. Model Development Challenges

The effort involved in building accurate spacecraft models of complex failure space are known to be a challenge, with high potential for model development and verification efforts to overcome the value of the model for fault management analysis. The project focused on two key methods to assure that modeling costs do not overwhelm the benefits of the fault models:

- Use of a model for multiple analyses, design products and operational fault management.
- Automated or semi-automated model development methods.

A. Multiple Model Uses

Throughout the systems engineering and design phases of a project, there are potentially numerous fault and failure related analyses that are performed. During operations, automated diagnosis, fault effects assessments, and recovery planning and execution can all be supported or assisted by the fault model. Even though different analyses and operational needs depend on very similar information, there is often considerable duplication that increases the overall effort or reduces effectiveness, as shown in Figure 1. Programs may choose to perform analysis using design data in

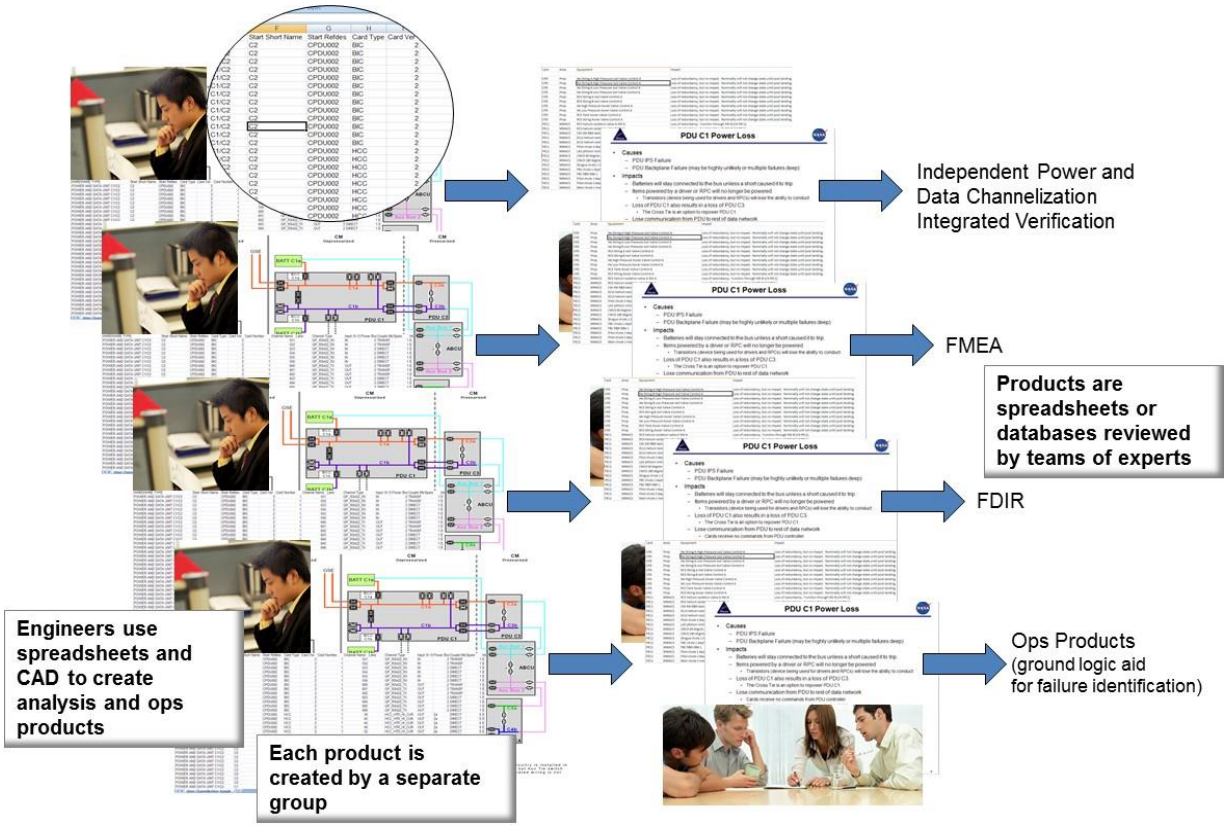

Figure 1 – System Engineering and analysis often relies on multiple teams using low level design data to create products, duplicating the effort of extracting and organizing information for each product.

the form of spreadsheets, databases, text documents and drawings, with each analysis pulling from the same set of sources, duplicating the research effort needed to obtain the information. If one analysis group considers creating a model that helps to clarify the inputs and automate the search for analysis results, the cost of developing the model compared with the "brute force" method using raw data can be considerably more than the effort for the analysis.

However, when a model serves the needs of multiple analyses and also supports operations, the model-based approach will generally provide value by reducing the overall effort. The Orion Program selected a channelization analysis as a target for demonstration of model-based fault analysis as a focal point for the ACAWS project. Channelization Analysis includes assessing the connectivity of the system to determine if there are single-point faults that can lead to loss of critical function. Orion developed a channelization database containing all the point-to-point connections between components. Manually, the analysis required that a team of domain experts review spreadsheet reports extracted from the database, essentially tracing from one row to another to find the paths from a power or data source to the loads at the end of the power channel. Then the analysis asked what would be affected if the power channel were to be cut at various points, representing a failure of the component next to the channel hypothesized to be cut. If the result was loss of critical functionality, the analysis flagged a design issue. In addition to loss of power, some failures can result in loss of sensor data that is required for fault responses. Orion design policy was that any FDIR algorithm that did not have valid data on all inputs would be inhibited from any execution, so that failures resulting in loss of data could preclude the very recovery software intended to recover from the fault. Finally, incorrect sensor data, such as an off-scale indication, possibly caused by some other failure, could cause incorrect execution of FDIR. The channelization analysis needed to assess all of these conditions.

Although the project focused on the channelization analysis, the Orion Program conducts numerous fault management analysis activities during the design life cycle that draw on the similar resources. As shown in Figure 2, using a fault model to organize design data inputs will reduce the effort and improve the quality of the analysis and operations products analysis needed by program fault management.

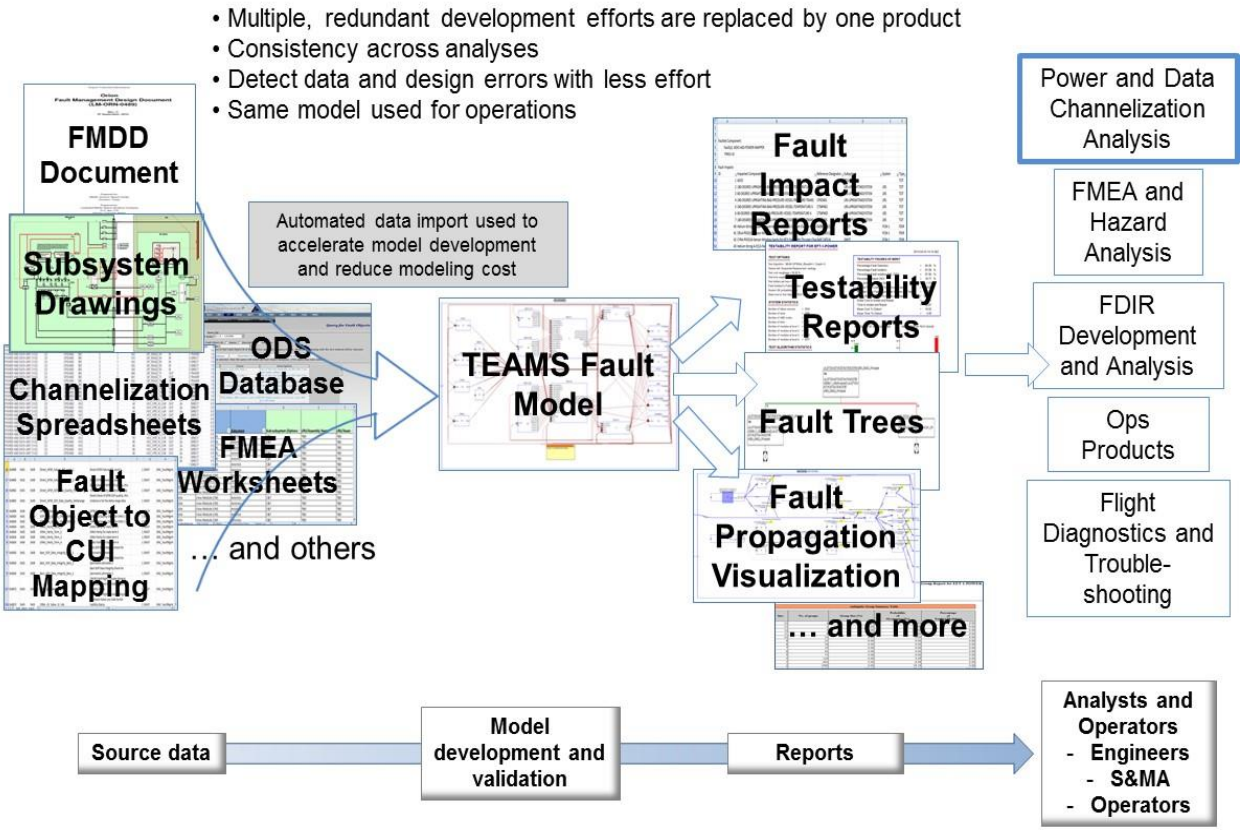

Figure 2. A Fault Model organizes system data one time to be used for multiple analysis, design product and operations product development.

B. Automated Data Import

The data required for development of a fault model can exist in a wide range of forms and styles, varying extensively between flight programs (Orion and Space Launch System, for example) but also varying between subsystems within a program. Different design organizations make different choices about how to represent their design, and different subsystems have different information needs; electrical power systems information can be significantly different from fluid or mechanical systems. This variability presents significant challenges to developing methods to automate the import of data. However, the goal of automated import of data is recognized as key to efficient and effective use of fault models, since manual data input for both initial development and maintaining synchronization with inevitable design change can overwhelm the modeling effort.

The ACAWS project addressed these challenges by assessing the data available from the Orion Program, selecting data sources that contained information required by the model and were amenable to automated processing, and applying tools previously developed and used by NASA Ames Research Center for importing into a model[6]. As with any program, design data comes in many forms, from Computer Aided Design (CAD) drawings, word processor documents, PowerPoint slides, databases, spreadsheets and numerous other formats. The Orion Program used a channelization database containing the electrical equipment information and the primary wiring segments and connectors between components, which constituted the power and data channels in the spacecraft. The component and connection information in the channelization database also forms the core of a fault model. The Orion channelization database could be readily exported to comma-separated value (csv) format which was readily processed by the existing tools after some filtering, checking and cleaning. Although it is a "power-assisted" methodology rather than a fully automated model creation process, the time and cost reductions are considerable. However the created model does not necessarily result in a visually readable model when viewed using the TEAMS Designer tool. Nevertheless, it proved its value in quickly building up a model populated with components, connections and failure modes.

The channelization data did not include the system's failure modes. These were obtained from Orion Failure Modes and Effects Analysis (FMEA). The process illustrated in Figure 3 shows the general flow that was developed to create data files that could be imported into the TEAMS model. The FMEA to Channelization mapping challenges were compounded since the FMEA did not have a complete set of items that matched exactly to the Channelization Connections Report. Therefore, the challenge was to map all possible FMEA items and failure modes to determine the best match with a Channelization End Component type. Where failure modes were not available or at an unusable level, the FMEA was further developed to a more complete state to fulfill the specific failure modes required by the

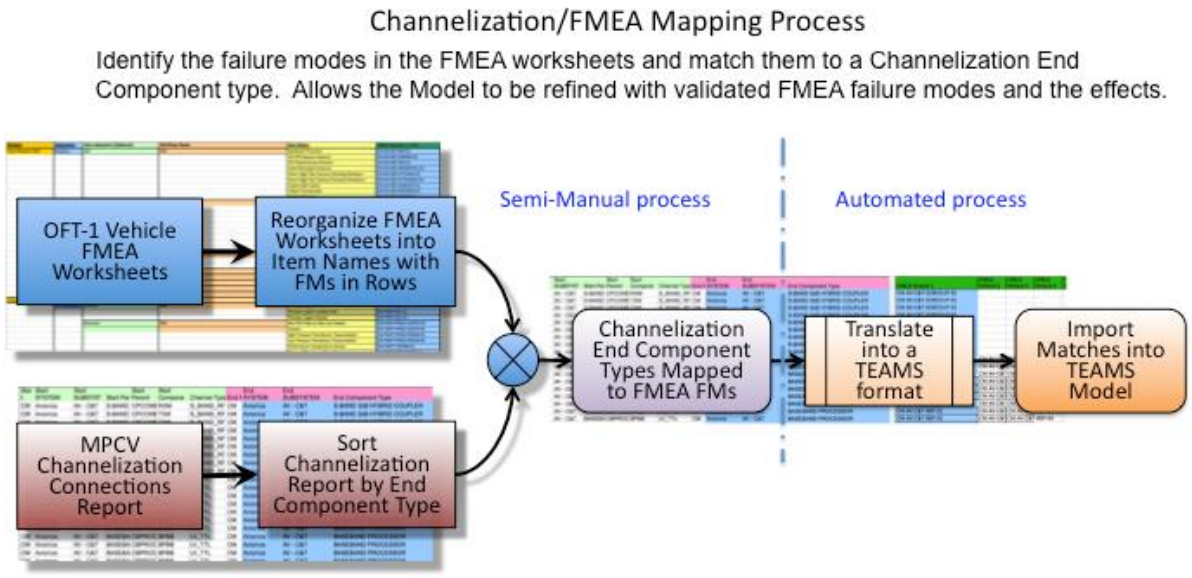

Figure 3. Process Flow to identify the failure modes in the FMEA worksheets and match them to a Channelization End Component type for automated import into the TEAMS Model

model. Utilizing several different data files as part of the FMEA worksheets and the channelization reports, the development of special processes were put in place to re-arrange, filter, and to match the two different data resources into something that fulfilled the TEAMS model generation input requirements.

Given the large range of possible formats of design information, an automated process that can handle any type of data was well beyond the scope of the project. Either modifying the data to fit the tools, or modifying the tools to fit the data format, could significantly enhance the ability to automate model development and interchange of data between models. Some aspects of both were done to adapt previously developed import tools to the Orion channelization data. Use of more standardized design tools and languages would significantly simplify the challenges of fault model development. The System Modeling Language (SysML) is an emerging design language that would provide a standard import source if used broadly across a spacecraft development program. SysML is a broad language with many choices about modeling methods and practices, so even programs using SysML will not necessarily generate SysML models that are compatible with TEAMS model auto-creation tools. SysML would, however, provide a much more contained range than the open-ended set of design tools and methods in use today. To explore how SysML could be used with TEAMS fault models, the project conducted a small-scale assessment of importing from SysML into the TEAMS model. A group at JSC has conducted studies and advocacy for broader SysML usage across NASA programs. This group developed a SysML model of a battery, containing multiple cells, sensors, internal circuitry and controls. Using a MagicDraw tool plugin, information similar to the channelization data previously imported was generated from the SysML sample model, demonstrating a relatively quick and simple method for converting SysML to a TEAMS fault model, as shown in Figure 4. The method required that the information was modeled in SysML constructs recognized by the plugin, but the plugin was fairly simple and could easily be modified to adapt to other modeling choices. It proved to be a simple method for modeling in SysML and automating the transfer of information to the fault model. The TEAMS Designer vendor, Qualtech Systems, Inc. is currently investigating

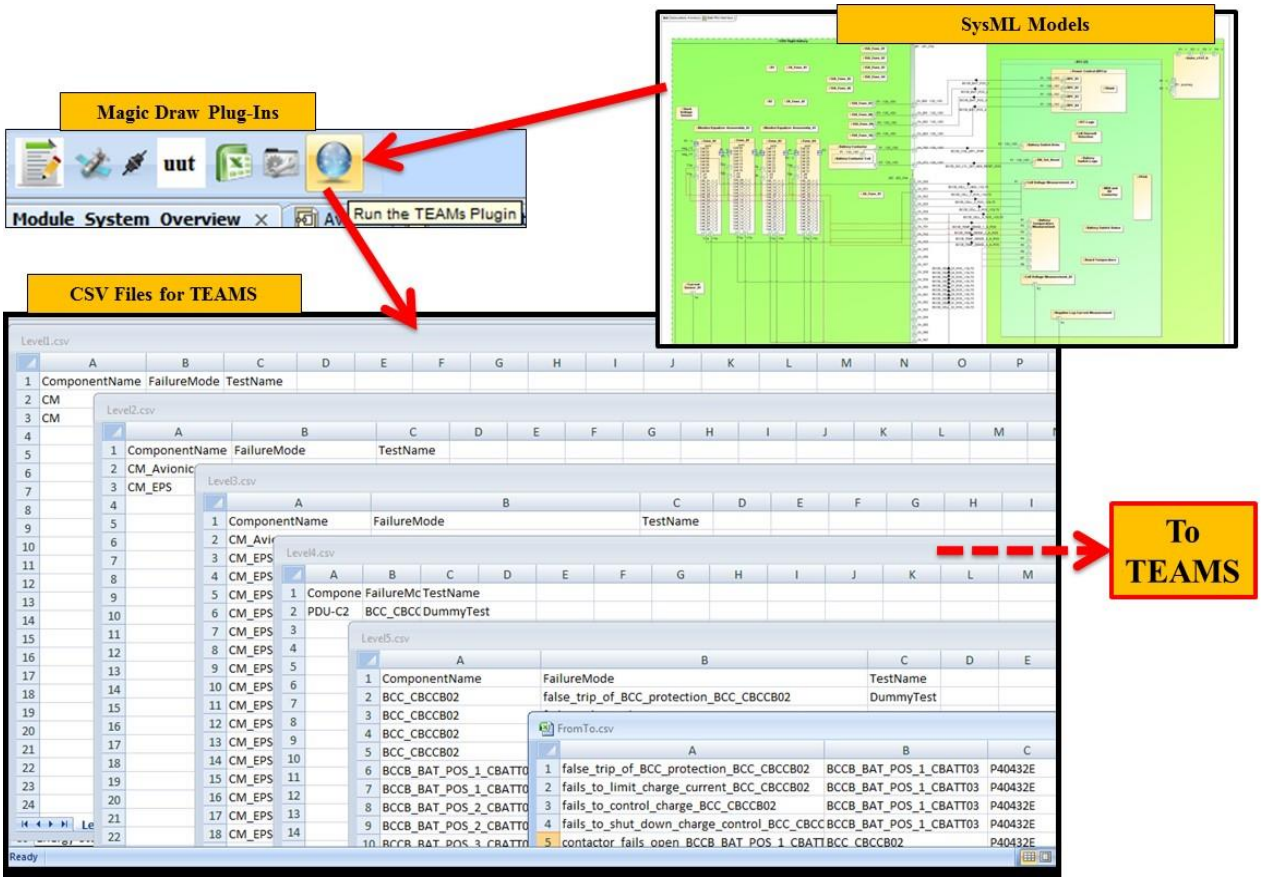

Figure 4 – SysML models can contain much of the information required for fault models. *The information can be automatically extracted and imported into the TEAMS fault model.*

more advanced and comprehensive methods for information transfer between SysML and fault models relying on the XML representations of SysML models and the XML schemas used for TEAMS fault models. With robust model transformation tools, the modeling effort can be substantially reduced. Use of SysML as a standard Systems Engineering practice that is focused on system design (i.e., "success space") with compatible tools for capturing and analyzing failure space information holds considerable promise for improved processes encompassing both design and fault management.

IV. Use in Fault Management Systems Engineering

Using the fault model developed by the ACAWS project, reports were generated that directly identified the loss of function and loss of redundancy that would result from each failure mode in the model. The original EFT-1 channelization analysis had already been conducted before the ACAWS fault model was developed.

Several of the analysts who had completed the channelization analysis were available to assess the TEAMS fault model for its effectiveness in simplifying the channelization analysis using model features for fault propagation from failure source to end effects. The Orion Program analysts had no experience in working with TEAMS models, so sessions were conducted with the ACAWS modelers and Orion Program analysts in which reports could be generated in response to analysts' questions. In each session, the Orion analysts selected fault scenarios of interest, reviewed the outputs from the model and used the information to complete a standard template consisting of failure, impact and work-around. The ACAWS modelers navigated the model, presented the model's outputs in either report formats or graphical model views depending on the Orion analyst requests. The model's outputs were very effectively used by the Orion analysts to determine the impacts of failures. Failure impacts included both loss of function and loss of redundancy. The primary purpose of the Orion channelization analysis was to determine if there were any cases in which single-point failures result in loss of critical function. The analysis was conducted after program Preliminary Design Review and before Critical Design Review. Any cases in which loss of function could result from a single failure would be design errors, and the program would need to determine if the probability and consequences of the failure occurring would warrant design modification. By the time we conducted the analysis exercise, no additional such design errors were found. Although it is a difficult task to prove that there are no undetected design errors, the analysts concluded that using the model would clearly provide them with the information needed to detect design errors, and the absence of any provided confidence that the connectivity was well designed. Although they were able to complete the analysis using primary data sources such as CAD and spreadsheets, the assessment of the model's value indicated considerable improvement. Some of the post-exercise observations were:

- Analysis using primary source data would take about 40 hours, but doing the same analysis with the model's output reports could be done in about 4 hours.
- The model captured cross-subsystem or cross-discipline failure effects that were not present in any one schematic diagram.

To complete the assessment of a model-based analysis compared with an analysis using primary source data, it is necessary to include the effort involved in building the model. Estimating modeling effort is not particularly straightforward based on a technology research and development project. Modelers had considerable experience building fault models on other projects, but did not have prior systems knowledge of Orion. In addition to model development, modelers were adapting previously started automated import tools to the Orion data. As a small-scale project and not tasked with rigorous model requirements for scope, accuracy and certification, the team worked toward soft goals for scope and accuracy but with firm completion dates. Assuming that the tools and processes were mature and familiar to the modeling group, the Orion model could be built with good assurance that it captured the primary design data correctly and was thus an accurate representation. However, the primary design data was not necessarily guaranteed to be accurate. We found during the course of the model development and analysis that data contained occasional minor errors, such as duplications, missing data or inconsistencies between related data. The rigor required to import data into a model requires filtering, formatting and checking the data in ways that could be overlooked by analysts reviewing the data. Some minor errors might be innocuous to a human analyst, such as a mismatch between upper or lower case that the analyst could readily conclude meant the same thing. Our model development work did uncover some data inconsistencies that had escaped the attention of program designers and analysts through design reviews and prior analysis, although they did not change the primary analysis results of the effects of faults on system functionality.

The model development and analysis exercise provided some insight into a model-based design analysis process. A highly automated, model-based analysis whose results would be used directly to confirm a design quality would require a complete, well-tested model. As a model-based assistance to a team of expert analysts, the model may not

require as much rigor, but would then require the analysts to confirm the results through other sources. In this scenario, the analyst could review the model's results, and use them as a guide to check the CAD and spreadsheets.

V. ACAWS Execution Architecture

The ACAWS execution architecture for EFT-1 was developed with technology developed under previous technology development programs[7] and adapted for use with EFT-1 data and models. Software was designed to execute on the ground using the EFT-1 telemetry downlink during the mission, with provisions for playback of flight and test data before and after the mission. The model originally developed for use in design analysis formed the basis for run-time health state determination during the mission. Health state determination included both diagnosis of faults and the determination of the effects of faults on other system components. Both loss of function and loss of redundancy due to a fault were determined. Displays were developed to show the diagnosed faults, the components affected by faults, and associated configurations and the actions of on-board FDIR related to diagnosed faults.

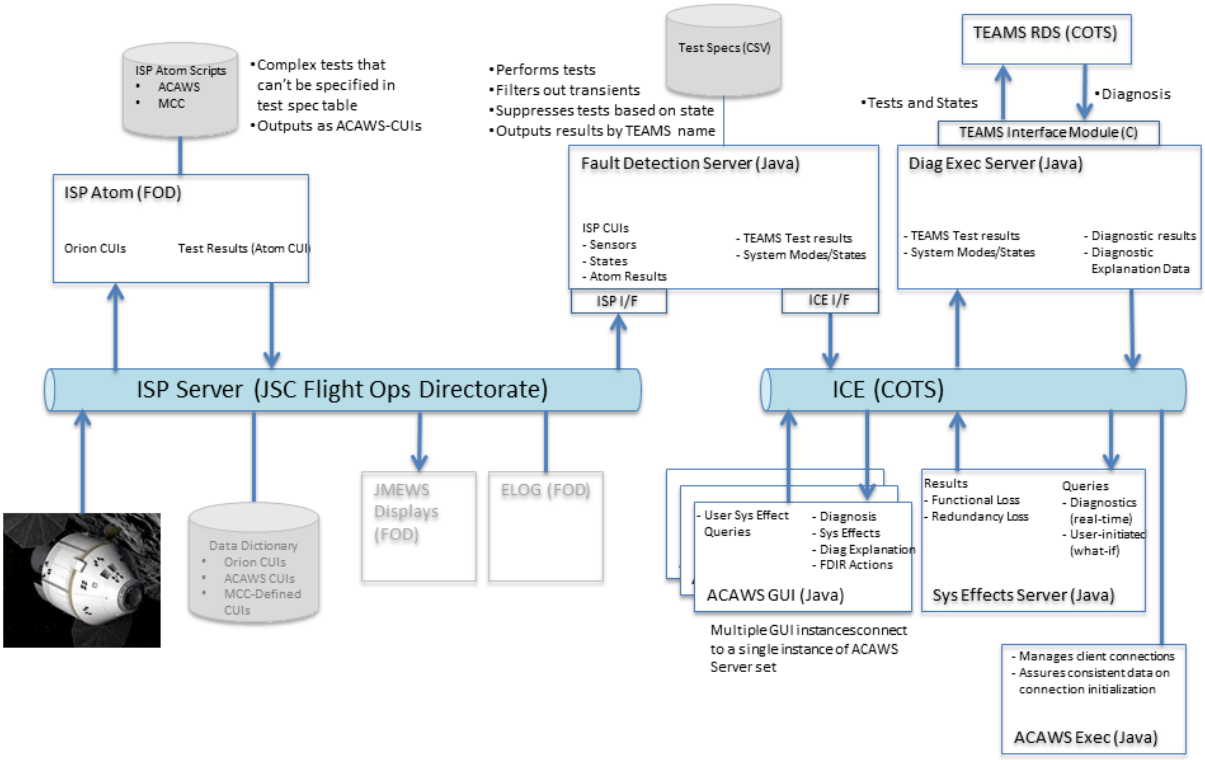

Figure 5. The ACAWS architecture. *ACAWS receives data from the spacecraft telemetry, performs validation and tests of the data in the Fault Detector, executes a Diagnostic reasoner and System Effects reasoner to determine the system health state and distributes information to multiple users.*

ACAWS performed health state determination in a multi-tiered architecture, diagrammed in Figure 5. The tools and applications used for the test, evaluation and flight included:

- TEAMS-RDS – A commercial product that performs model-based diagnosis, using the test results from the Fault Detector;
- Telemetry Interface – the reception of data from the Orion spacecraft via NASA Mission Control Systems data distribution network;
- Fault Detection - Includes data intake from the downlinked telemetry, data cleaning and validation, and testing for indications of faults yielding PASS, FAIL or UNKNOWN test results;
- Diagnostic Executive – Manages the interface to the diagnostic reasoner and distributes results to the System Effects Reasoner and Graphical User Interface, logs data;
- System Effects Reasoner – Determines the effects, or impacts, on system components due to a diagnosed fault;

- Graphical User Interface - Distribution and display of health state information;
- Mission Control Center Applications – some of the key tools used by flight controllers during the EFT-1 mission were used for telemetry displays of control and health information. They provided a realistic flight control environment and a basis for comparison between baseline flight control applications and with the addition of ACAWS displays.

Each of these will be described in the following paragraphs. The TEAMS-RDS diagnostic reasoner and its use of the TEAMS model is described first, since it provides the core of fault diagnosis, and the other architecture elements listed above either support or depend on the TEAMS-RDS diagnosis.

C. TEAMS-RDS

The TEAMS fault model is a system dependency model that includes system components, their failure modes, connections between the modules, and test points that are used to detect indications of faults[8]. It also includes mechanisms to represent redundancy and configurations, such as ON and OFF states of equipment. The modeling tool, TEAMS Designer, is used to build a graphical model, and the tool then generates the run-time representations used by the TEAMS-RDS reasoner to perform fault diagnostics. Connections between modules represent the propagation paths of faults. The paths usually represent a physical path such as a wire or pipe, but could represent physical proximity such as a heat-sensitive component near a source of heat. The failure modes of components contain one or more functions affected by the failure, and tests in the model detect the presence or absence of these functions as they propagate (or fail to propagate) along the connection paths in the model. A test may fail because of any number of faults. Figure 6 shows a hypothetical Power Distribution Unit (PDU) providing power to a Pump, with internal components of the PDU that are viewed by expanding the PDU to open the next level in the hierarchical model. Tests include a Pump Rotation test, such as an RPM measurement at the pump, and power measurements at the PDU Bus and at the output of a Remote Power Controller (RPC) (basically a power switch). In the example, the Pump Rotation test could fail because of a mechanical fault of the pump, or because one of the failure modes in the Power Distribution Unit resulted in loss of power to the pump. If the PDU Bus Power test and PDU RPC Power test both pass, the reasoner exonerates the PDU components and concludes that the Pump has failed. If both PDU Bus Power and PDU RPC

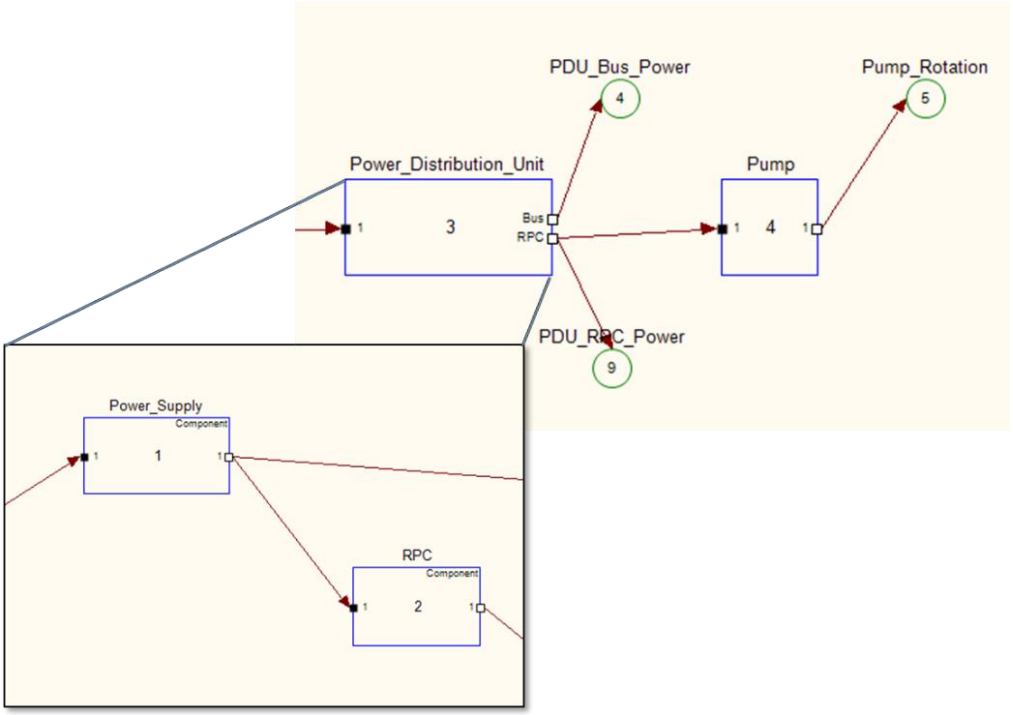

Figure 6. A fragment of a TEAMS model. *A hypothetical power distribution component illustrates components, tests and propagation paths. The Power Distribution Unit is expanded to show its internal components, a Power Supply that provides internal power to the PDU and an RPC (switch) that controls power to the Pump.*

Power tests fail, the reasoner determines that the PDU Power Supply has failed since it is the single failure that results in all three tests failing.

The graphical model is reduced to a matrix of the tests and failure modes, known as a D-Matrix (from "dependency matrix") for run-time diagnostics. A fragment of a D-Matrix that corresponds to the model fragment described above is shown in Figure 7.

	PDU_Bus_Power	Pump_Rotation	PDU_RPC_Power
Pump.Mechanical_Failure		1	
Power_Supply.Fail_to_Gen_Power	1	1	1
RPC.Fail_Open		1	1

Figure 7. A D-Matrix correlated to the model fragment in Figure 6 is the key to run-time diagnosis. *A '1' indicates that the test in the column heading detects the failure mode in the rows. The diagnostic reasoner uses the test results provided to the D-Matrix to determine the root cause that best explains the complete fault signature.*

D. Telemetry Interface

The Orion telemetry was transmitted from the EFT-1 spacecraft to the Mission Control Center (MCC) at JSC via various data links from pre-launch through post-landing, and ultimately converted and distributed within the MCC using the data distribution architecture that has been used for Shuttle and ISS data known as Information Sharing Protocol (ISP). The ACAWS project established a connection to the MCC ISP using secure data protocols and configured in a receive-only mode to assure that inadvertent data could not be inserted into the flight data stream. Once in the ACAWS test lab, the data was distributed to the ACAWS applications as well as to copies of the MCC data displays so that the team could see what the flight controllers were using, alongside the ACAWS diagnostics and effects displays. ACAWS diagnostic applications subscribed to about 2,000 data elements, while the MCC displays in the ACAWS lab processed roughly 18,000 unique data elements.

ISP was operated in a "change-only" mode so new data is received only when the downlinked value changes. The capability is sufficiently robust, brought on line roughly mid-life of the Shuttle program and in use for all of ISS, to make a valid assumption that no change in value or status meant that the data had been downlinked with the same value as the last value transmitted by ISP.

Data rates of the downlinked data ranged from 40 Hz to 0.1 Hz, but ISP provided data packets at 1 Hz. For data downlinked at rates higher than 1 Hz, all changed values would be received as a block of data. For example, a 10 Hz data element may downlink 6 values that had changed from the previous data element, and 4 values that were the same as the previous. ISP would deliver the 6 changed values, but it could not be determined which values in the sequence had changed. ACAWS simplified its use of higher rate data by using only the last value received. This simplification did not appear to affect any of the failure modes for which the model was configured.

For data at rates slower than 1 Hz, the absence of data in a cycle meant that either the value was not in the downlink in that 1 second cycle, or it had been received but was the same as the previous downlinked element, since ISP does not provide frame boundary information.

E. Fault Detection

The first step in performing fault diagnosis was to sample the downlinked data and perform tests on the data values. Tests could evaluate to PASS, FAIL or UNKNOWN, and the vector of test results were sent to the TEAMS-RDS diagnostic reasoner. The Fault Detector subscribed to ISP data and performed the data testing needed for diagnosis, after some pre-processing to filter and frame the data.

Prior to testing for a threshold value, the data was first tested for a variety of possible validity errors. The MCC data interface system reported if the data was static, such as in planned or unplanned Loss of Signal (LOS) periods and if so, no further testing was performed and test results on the data were set to UNKNOWN. In addition, ACAWS pre-processed the incoming data to determine if a value was off-scale low or high, which could indicate a sensor failure, analog-to-digital input/output (ADIO) conversion problem or other data transmission failure. Data was also tested to determine if it was not changing as often as expected. Most sensor values would change periodically even if the measured value was quite stable, so if a change was not observed for several data cycles the value was tagged as "flat-lined" (to differentiate from a static indication in the downlink) and the value would not be tested. The method was effective in differentiating between sensor faults and faults in a functional component. The UNKNOWN test result does not contribute to a diagnosis, so a false positive will not normally result from the flat-line filter, but was

effective at removing possibly suspicious data from diagnoses. If the data really was nominal but just exceptionally stable, diagnosis would be completed using any other tests, and the worst case would be that a component's fault state would be UNKNOW until data resumed updating. Without the flat-line filter, a sensor that had stopped updating would continue to yield a PASS result, and if a component failure occurred, the old and non-changing data might incorrectly exonerate the fault. Flat-line tests generally were not used individually to diagnose faults, but there were some failures that were characterized by whole groups of sensors freezing concurrently at their last value.

The relations between sensor, data transmission and component faults are illustrated in Figure 8. Consider a pair of redundant temperature sensors that are used to detect a battery over-temperature fault. Component failures are in boxes numbered 1 through 4, and tests are represented by the circles numbered 1 through 6 in the figure. Each sensor (box 3 and 4) could fail with an off-scale reading, or an ADIO converter failure (box 2) could result in both sensors flat-lining at their last in-range value. If either invalid case is detected, the ACAWS Fault Detector flags the sensor value as off-scale or flat-line, otherwise the values are valid. If either sensor is flagged as invalid, the Fault Detector sends UNKNOWN to the corresponding temperature tests (circle 1 or 2) that are used to detect a battery over-temperature fault (box 1). Suppose the Fault Detector identifies an off-scale value on Sensor A data and sends a FAIL result to the Sensor Offscale test (circle 3). Because the value is invalid, Fault Detector sends UNKNOWN to Temperature A test (circle 1). TEAMS-RDS will diagnose the Battery Temp Sensor A as failed. If Temperature B test result is PASS, TEAMS-RDS will determine that the Battery Overtemp fault is not present and the battery is still nominal, but now it must base its determination on a single sensor. Now suppose that the Battery ADIO converter fails and Fault Detector identifies both Temp Sensor A and B data as flat-lined. Tests 4 and 6 both receive a FAIL value, and TEAMS-RDS diagnoses the Battery ADIO (box 2) as FAULTED. The Fault Detector now sends UNKNOWN test results to both temperature tests (circles 1 and 2) and the Battery Overtemp fault (box 1) is undetectable because all tests that could detect the fault are unavailable. Furthermore, if the sensor data is flat-lined, it is no longer possible to know if the actual data has gone to off-scale, so the tests for off-scale (tests 3 and 5) should also be UNKOWN, and the Temp Sensor failure modes (boxes 3 and 4) are undetectable.

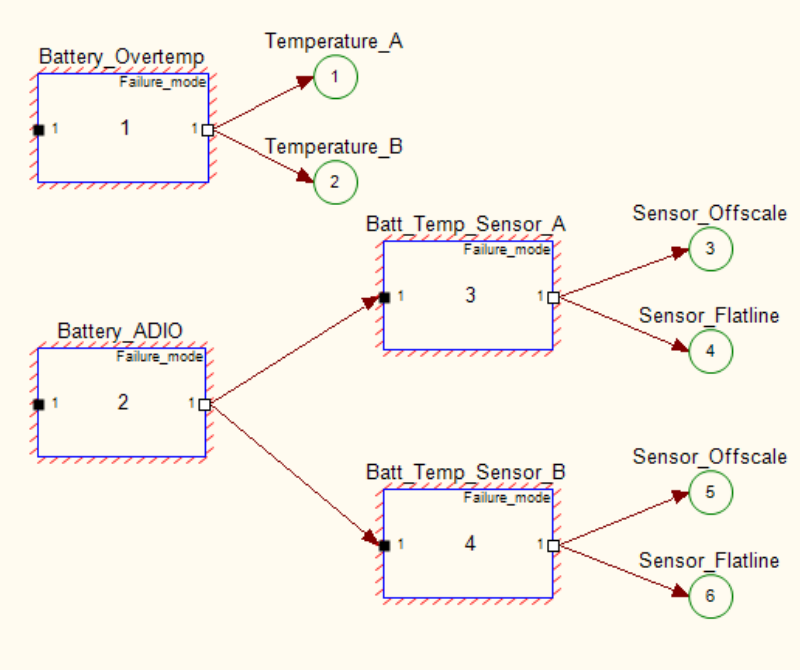

Figure 8. Tests on data values are correlated with tests of the sensor validity. *Tests are indicated in circles, and component failure modes in boxes. The Fault Detector evaluates telemetry data and sends PASS, FAIL or UNKNOWN to each test based on data tables that define the test conditions. Tests 1 and 2 are data value tests, while test 3 through 6 are validity tests. When validity tests FAIL, corresponding value tests using the data are UNKNOWN. In addition to suppressing data value tests, validity tests can be used for diagnosing failures of sensors and data processing components.*

Pre-processing also included the use of test suppression based on system configuration or prior faults. If a component was turned off, for example, the tests on its temperatures or voltages would not be performed. Data that reported the system configurations were tested and rather than being sent to the TEAMS model, test suppression logic was used that would override the computed test results and send an UNKNOWN result to the TEAMS-RDS reasoner instead. This method was effective in filtering tests once a failure had occurred, as well as preventing diagnosis on equipment that was not in use. For example, a backup coolant pump would not be tested for pressure output when it was turned off, so its tests would be suppressed until it was turned on.

The Fault Detection application performed a data framing function for telemetry that was downlinked at 0.1 Hz. This was needed to assure that test

results were only sent to TEAMS-RDS when data was received, and UNKNOWN results on the remaining cycles. Without the framing, a slow rate value could indicate a PASS for a few cycles after a fault occurred, which would exonerate the actual fault. When received from ISP, it was not known if the absence of a new value meant that a value was not downlinked in that particular 1-second data cycle or if it was received but had the same value as the previously downlinked data element. To frame the data, the Fault Detector maintained a counter on each of the data values with a downlink rate slower than 1 Hz. After 10 cycles, the value would be evaluated as new data and a test result would be evaluated and the counter was reset. If a new value was received before the counter expired, it would be evaluated and the counter reset. Since ISP was not rigorously synchronized to on-board clocks, the data occasionally updated a cycle earlier or later than expected, and this technique kept the data well-framed, sufficient for the types of failures for which the model was built to detect.

Once the pre-processing was completed, tests on data values were performed. Tests on telemetry values ranged from simple limit or range checks and equalities, to somewhat more complex tests involving multiple telemetry values. The simple tests were specified in a comma-separated values format, with a comparison operator that took one value, or a range operator with two values. Comparison operators included equal, not equal, less than or equal, less than, greater than or equal, and greater than. Range tests included an in-range test, that is, a test that fails if the value is between the lower and upper bound, and an outside-of-range test that fails if the value is at or outside of the lower and upper bounds.

For more complex conditions, a tool developed by NASA for flight controller computations on telemetry, ISP Advanced Tool of Math (ATOM), was used. ISP ATOM is a simple interpreted scripting language that allows for arithmetic and logical expressions, and storage of variables from one execution cycle to another. ISP ATOM reads telemetry values directly from ISP and publishes results back to ISP as unique data elements. The Fault Detector received the ATOM outputs and processed them as any other telemetry element. The ATOM scripts used by ACAWS always generated a PASS, FAIL or UNKNOWN result for consumption by the TEAMS-RDS reasoner. Since ISP ATOM operated before the ACAWS Fault Detector obtained the data, the ACAWS pre-processing values were not available to ATOM. This required an engineering trade-off for the more complex tests for which ATOM was well suited. On one hand, the Fault Detector pre-processor could easily be configured by providing a table of off-scale high and low values for each telemetry item, and a flat-line cycle count to specify how many unchanged values would be observed before setting the flat-line status. ATOM would have required several lines of scripting code to perform the same functions, so it was not considered viable to use ATOM for all telemetry validity testing. The options were to perform similar off-scale and flat-line testing of the telemetry used in the ATOM script before the test logic on the telemetry values, which made scripts significantly more complex, or use the values without additional validity testing, which could have resulted in incorrect test result if a sensor failure resulted in an invalid value. Generally the tests were used without the added validity testing. No problems were encountered during testing or the flight due to this simplification, although a robust flight system would probably need to avoid this sort of simplification.

F. Diagnostic Executive

The Diagnostic Executive received a vector of test results from Fault Detector and packaged them for transmission to the TEAMS-RDS diagnostic reasoner, and received the results from the TEAMS-RDS reasoner and distributed the results. Diagnostic Executive formatted the test results from Fault Detector and transmitted them to TEAMS-RDS once per second. TEAMS-RDS executed as a separate process with a TCP/IP interface. It was designed to be able to execute on a remote machine, although for the EFT-1 flight, RDS executed on the same machine as the ACAWS Diagnostic Executive, Fault Detector, System Effects Reasoner and ISP server.

When diagnostic results were received from TEAMS-RDS, they were transmitted to the Graphical User Interface (GUI), along with telemetry data used to provide an explanation of the diagnosis, and configuration information for equipment with switches or other on/off controls. The test lab used up to five copies of the ACAWS GUI, there was no defined limit on the number of GUI connections permissible.

Diagnostic results were also sent to the System Effects Reasoner each time the diagnosis changed from the previous cycle. The System Effects Reasoner transmitted results to the GUIs and did not respond back to the Diagnostic Executive.

An adjunct application to the Diagnostic Executive was used to buffer data so that GUI clients that were started after the Diagnostic Executive or System Effects Reasoner would receive the current health state data. Since diagnostics and effects results were sent when changes were made, a late-joining client would otherwise miss failures that occurred previously and were still present.

G. System Effects Reasoner

A System Effects Reasoner was developed at NASA Ames Research Center to determine the effects of a fault on other system components[9]. The System Effects Reasoner operates on an XML representation of the same TEAMS fault model as the diagnostic reasoner uses, with a graph traversal algorithm to trace a fault from the origin to all affected elements, as shown in Figure 9. The graph follows the connections in the TEAMS model, using the same propagation paths used for diagnosis. Whereas diagnosis traces backwards from tests to the source of the TEAMS functions to converge on the root cause fault, System Effects Reasoner traced forward on the comparable paths, fanning out from the diagnosed fault to all the components affected by the fault. The System Effects Reasoner used the test points as the end nodes of the graph traversal. The diagnostic test nodes were used by the System Effects Reasoner, but they were not always sufficient to determine all the effects. Diagnosis could be done by sampling data downstream of the fault without testing every data value that was affected by the fault. To identify all the affected components, however, every propagation path had to be explicitly modeled. For System Effects, extra test nodes not required for diagnosis were added that were labeled as "impact points". The TEAMS Designer features for creating the D-matrix and other run-time diagnostics data were able to use the test labels to exclude the test points used only for effects modeling. In essence, the TEAMS model included both a diagnostics model and an effects model, with substantial overlap between the two.

The EFT-1 Orion spacecraft, as do all modern spacecraft, included extensive redundancy. Fault effects include both loss of function and loss of redundancy. Loss of function occurs either as a result of a single fault for which no redundant component provides a necessary resource, or as a result of multiple faults causing loss of all legs of redundancy. When the resources required for a function are still present but now provided by fewer sources, the System Effects Reasoner reported a loss of redundancy condition. The ACAWS system was implemented to report loss of redundancy when redundancy was reduced to a single redundant path, that is, a single additional fault would cause loss of function. This choice was made in consultation with NASA flight controllers who typically must determine "next worst failures" after an initial failure occurs. Presenting only the components at risk of loss due to a single failure was most consistent with flight controller assessments of next worst failure, and displaying information about components that have gone from, for example, triple redundancy to dual redundancy, would have complicated

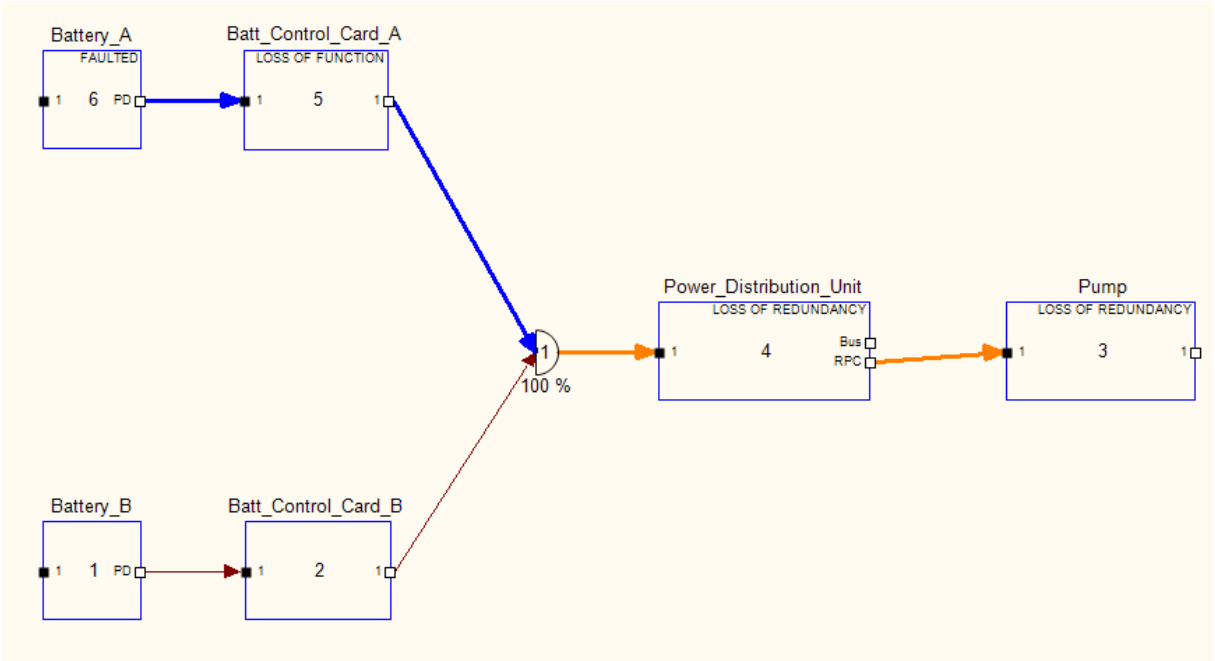

Figure 9. The System Effects Reasoner traverses the model to identify the effects of a fault. *The illustration starts from a faulted battery and follows paths to components that lose functionality due to the fault (blue line), and to components that lose a redundant resource (orange line). The AND node (semicircle labeled '1') does not represent a physical component, but indicates to the System Effects Reasoner where redundant paths join.*

displays and reduced the ability to see the components most at risk of loss. This was, however, a modeling choice that could be modified to include any reduction of redundancy.

H. Graphical User Interface

A Graphical User Interface (GU) was developed to display both real-time results and controller queries about possible or hypothetical faults, in a "what-if" mode of operations.

Health state information was displayed using icons attached to component and group annunciators. Some of the primary health state icons are shown in Figure 10, as they would be shown in the GUI annunciators, or display boxes labeled with a component or group name. Without an icon, the component is assumed to be operating normally and unaffected by any other fault. Health state information included:

- faulted components – a confirmed root cause fault;
- possibly failed components – an ambiguity group, used when a single fault cannot be isolated;
- affected components – components whose function has been lost or diminished by a fault in another component;
- possibly affected components – when diagnostic ambiguity exists, any components that would be affected by some but not all of the diagnostic ambiguity set are identified as possibly affected;
- configuration affected by FDIR – when the on-board FDIR takes actions to reconfigure the system in some fault conditions, ACAWS displayed the affected configurations, typically switches either turned off or on by FDIR;
- diagnostic explanation – the test algorithms, telemetry values and test results used to diagnose or exonerate a failure mode; this proved very valuable in providing confidence in the diagnostic results allowing operators to compare the ACAWS logic with their own extensive systems knowledge.

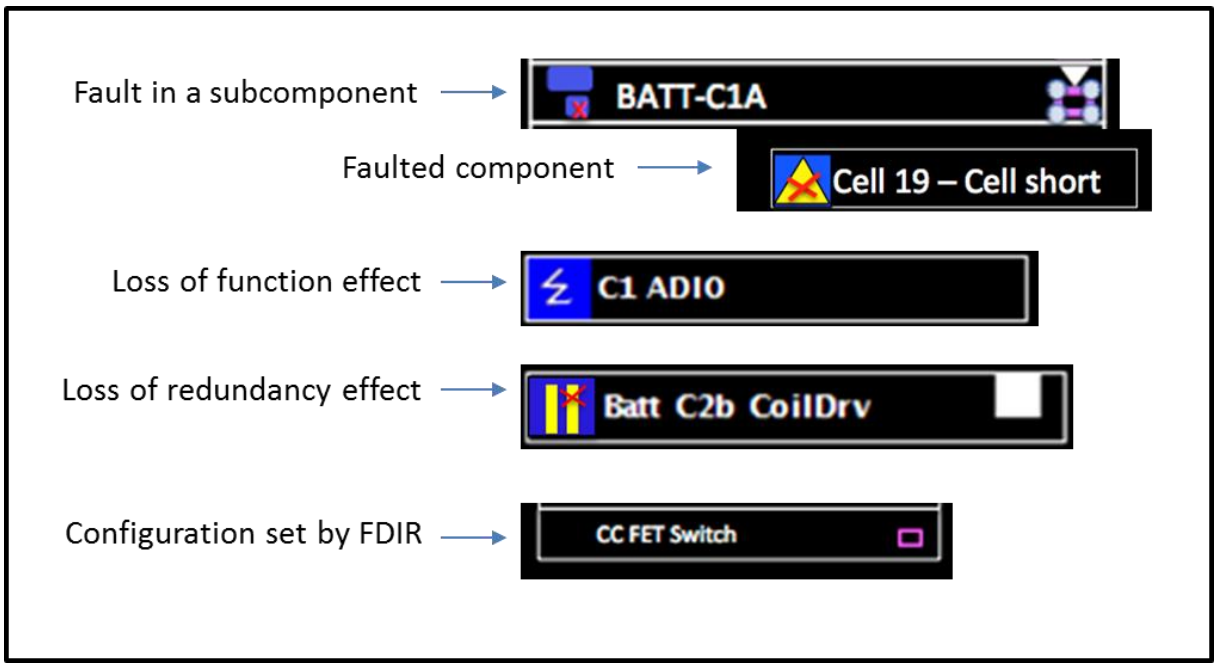

Figure 10 – Display icons were used to indicate faults and effects of faults. *Diagnostic and System Effects are shown in blue boxes at the left side of component annunciators. Boxes to the right side of annunciators indicate ON/OFF states. A filled box indicates ON, an open box indicates OFF, and magenta border indicates that the configuration was set by FDIR.*

In addition to the run-time results, several capabilities used the same model and logic to present operator queries, either in an off-line mode or in a hybrid mode[10]. In the off-line mode, real-time diagnoses would not be accepted, and only operator queries would be considered. The flight controller could specify one or more hypothetical failure modes and configurations and query the system to determine the components that would be affected if the specified failure modes had occurred. In the hybrid mode, the current real-time diagnoses would continue to be displayed and evaluated,

and the controller could determine what would be affected if other components were to subsequently fail. A direct "next worst failure" query was also available; after a failure, the components that had lost redundancy from the first failure could be selected to determine what single faults would cause loss of function of that component. Flight controllers could use these features to explore possible failure scenarios that help them to make operational decisions to avoid or plan for possible further failures.

The Orion spacecraft has thousands of components, with each component having multiple ways that each can fail. The quantity of information presents challenges to designing a display. In failure situations, operators need ready access to information, without having to traverse many screens to get to the information. The ACAWS displays used combinations of graphical schematic representations, annunciator panels with hierarchically organized components, and text displays listing diagnosed failures and possible failures. The goal was to get to the necessary information in at most three screen changes, and two changes whenever possible. Information roll-up was done in two forms – components with subcomponents, and groups of related components. The effect of a failure of a subcomponent on the functionality of the total component could vary considerably. For example, a power control card consists of several Remote Power Controller (RPC) subcomponents, and a failure of one would not affect the overall card's function other than the particular RPC, but a card controller or card power interface could fail resulting in total loss of function of the entire card. Total loss of component functionality was indicated as a loss of component functionality, whereas failures that affected only particular subcomponents were displayed using a "failure inside" icon, which required the operator to open another panel to see the particular failure mode. The other information roll-up used groups of similar components, such as the pyrotechnic initiators in the landing and recovery system. If any component in the group had failed, a failure icon was displayed on the group annunciator panel, and the operator would open another annunciator panel containing the members of the group.

I. Mission Control Center Applications

In addition to the ACAWS diagnostic elements described above, the demonstration and evaluation environment for ACAWS included the primary flight control displays used in the EFT-1 Mission Control Center. The displays are used to present telemetry data for each of the subsystems, including sensor data, configuration such as switch positions, FDIR data, pyro event sequences, mission phase and other operational and health data. The flight control displays connected to the same ISP server as the ACAWS applications used, allowing the ACAWS flight observers to see the same data that the EFT-1 flight control team used. The environment enabled a direct comparison between the information provided by current flight control tools and health state information provided by ACAWS. ACAWS displays were designed to augment the currently available information, not to replace existing displays that include both operational control and system health information. The MCC displays were used both for the flight demonstration and for an extensive evaluation conducted prior to flight, as will be described in the following section.

VI. System Testing and Evaluation

The system was tested extensively using nominal test data from the Orion Program and combined with fault signature data developed by the ACAWS test team and supported by NASA Flight Operations Directorate. Nominal data was recorded when Orion test data was transmitted to the Mission Control Center, either from the Orion EFT-1 spacecraft during vehicle checkout, or from Orion test and simulation systems. The ACAWS test team identified the telemetry that would be affected by failures and developed a fault specification methodology to overwrite the nominal data with the faulted data. For example, if a battery fault would be characterized by a low voltage and rising temperature, the sensor values and timing of the changes were specified, and a script would read the nominal data and replace the data assumed to be affected by the failure.

A series of failure scenarios were built up using these methods for a controlled evaluation of the ACAWS system with teams of NASA flight controllers, in a configuration designed to function as a flight control room mockup, as shown in Figure 11. The purpose of the evaluation was to validate design decisions made in ACAWS development, elicit additional requirements for future development, and to obtain objective evidence for the value of higher order fault management information than has been available for prior human space flight vehicles.

Flight controller participants were separated into two groups of four, where each group consisted of three controller positions and a Flight Director. Each group participated in two test sessions separated by two weeks to reduce the effect of learning on performance. Test sessions consisted of 8 scenarios, each 9 to 10 minutes duration, and containing two or three independent faults. The conditions were counterbalanced so that one group received the baseline condition first, using only the displays used by the EFT-1 Flight Control team for the actual flight, and the other group received

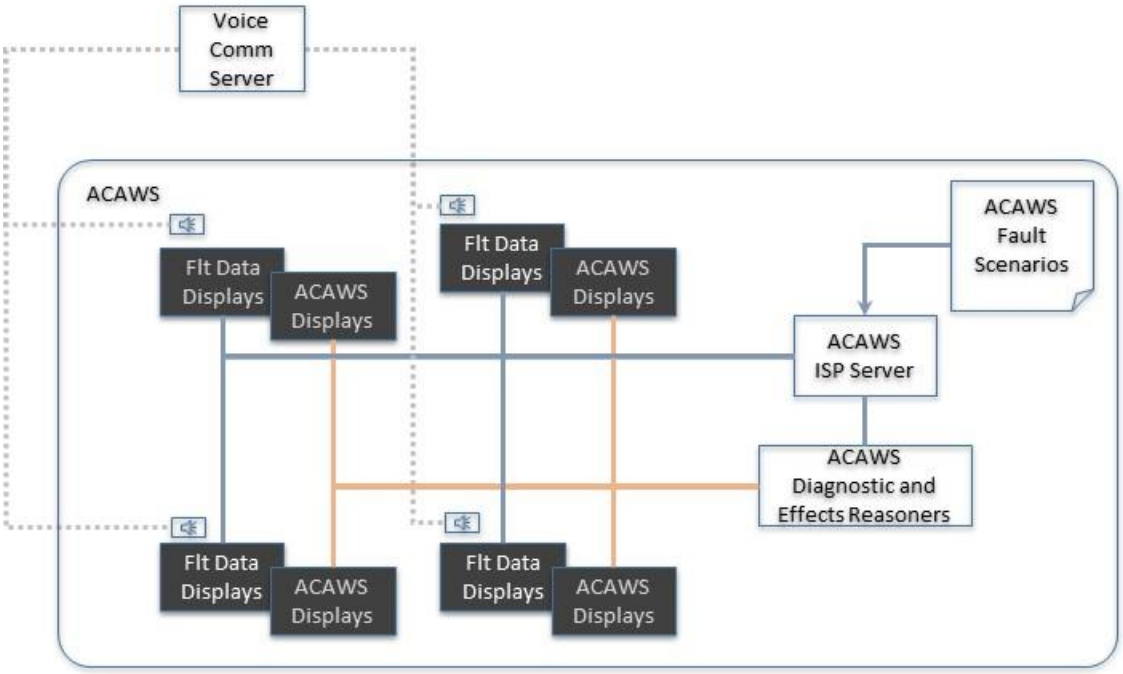

Figure 11. The test and evaluation environment. *The ISP Data Server played back fault scenarios constructed from recorded Orion test data with injected faults to the ACAWS reasoners and Flight Controller displays. Voice communication loops added operational realism and allowed for recording of discussions by evaluators.*

the baseline plus ACAWS condition first, using the ACAWS information and displays in addition to the EFT-1 displays. Communications loops were used to provide realistic communications between controllers and the flight director. All communications were recorded for use in later analysis, as were all cursor and keyboard movements and screen layouts. Evaluation questionnaires were completed for each scenario and also upon completion of each test session to obtain the subjective judgment of the flight controller evaluators regarding the strengths and weaknesses of the system as currently designed and what additions and modifications would be most beneficial.

During data analysis, it became apparent that both groups retained memory of the scenarios in the second run, which made it somewhat difficult to obtain pure objective comparisons. The teams had only brief training and familiarity with ACAWS displays, but extensive experience with the tools used for EFT-1 flight control display and some familiarity with Orion design, even though the teams were not the EFT-1 Flight Controllers. The data did not reveal a significant reduction in workload when using ACAWS. Both groups had more difficulty in the first session in diagnosing faults and making control decisions to respond to the faults regardless of ACAWS use. By the second session, they had become more familiar with the failures and were quicker to recognize and respond. Nevertheless, the flight controller inputs to the ACAWS evaluation indicated that the information provided by ACAWS would be an effective aid in rapidly understanding failures and responding more quickly and accurately with adequate training and familiarity with ACAWS.

The exercise was conducted to assess the information requirements for higher autonomy systems, and also to evaluate the ACAWS system as a ground operations tool. We coordinated closely with the NASA Astronaut Office group that is developing Orion on-board display requirements and prototype displays to demonstrate the quality of information available from the ACAWS system. Current focus of the display development group is on the first Orion crewed mission, but beyond EM-2 with potentially long-duration missions to an asteroid, the information that ACAWS demonstrated will be vital on-board information needed to meet the spacecraft autonomy requirements including unassisted handling of failures.

VII. EFT-1 Flight Results

The ACAWS system monitored the telemetry downlink from the Orion Exploration Flight Test 1 (EFT-1) mission in December 2014, from pre-launch through post-landing power-down. The first launch attempt resulted in a scrub due to high winds at the launch site and minor system issues. The second attempt on the following day culminated in a nominal launch at the opening of the launch window, Figure 12. The 4 ½ hour mission was nearly fault-free. ACAWS detected a single sensor failure, found to be a result of damage during vehicle assembly.

The ACAWS system was configured to detect and display approximately 3500 failure modes, ranging from sensor faults to major system component failures, monitoring approximately 2500 unique telemetry elements. The system executed on a Linux laptop at a 1 Hz execution rate.

Although a rigorous performance analysis was not conducted, the system appeared to handle the throughput with ease. Through the two days of monitoring the data, including about 3 ½ hours on the first attempt, and 6 hours during the successful mission, the ACAWS system received and processed all data without interruption or running behind the flight data.

As would be expected with the initial flight of a new spacecraft, there were notable differences between flight data and the test data used for system testing. Some of these differences included:

- Environments. The thermal environments, during pre-launch and throughout the mission, were considerably different than had been simulated. Prior to launch there were temperature sensors reporting temperatures outside of the nominal control ranges, which ACAWS interpreted as possible heater or power failures. Once in flight the heaters all behaved nominally and ACAWS cleared the heater faults, although the flight thermal environments varied considerably from the simulated environmental data used in testing.
- Configurations. Although testing generally accounted for all planned configurations, there were timing and sequencing variations that had not been observed by the test data used with ACAWS, notably in the post-landing power-down sequence. ACAWS diagnosed at least one fault in late post-landing power-down sequence that was due to components being turned off per the plan.
- Data dropouts. Although simulated test data included some loss of signal events, the actual flight was the first time that ACAWS had been exercised throughout the reentry blackout and with low data rates during ascent while the Launch Abort System was attached to the spacecraft. ACAWS handled the loss of data without false diagnoses due to loss of signal.
- Data variability. The degree of data variability was somewhat different from test data. Many of the measured values varied in flight by only one or two calibration counts (the raw sensor units after conversion from analog to digital, but prior to conversion to engineering units), whereas typical test data showed a little more jitter. This affected the "flat-line" pre-processor test to filter out possibly unresponsive sensors. The flat-line thresholds were set based on the Orion program test data. Although no false positives or false negatives were attributed to these variations, program maturation will need to improve these types of data filtering.

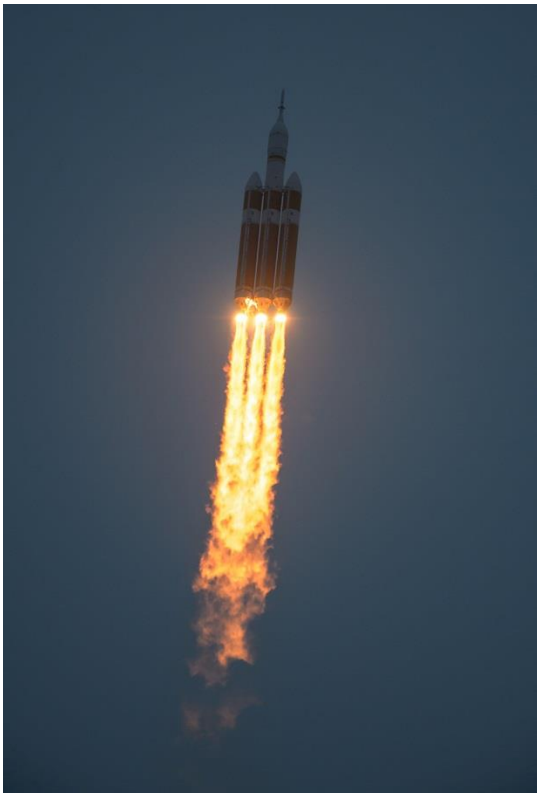

Figure 12. The Orion EFT-1 spacecraft begins its 4 ½ hour mission atop a Delta IV Heavy launch vehicle at Cape Canaveral, FL on December 5, 2014

Monitoring of the flight data commenced about 30 minutes prior to launch. Although the project objectives had not included pre-launch or post-landing system fault diagnostics, and very little effort was put into assessing the pre-launch data and environments, the opportunity was taken to connect and begin the evaluation before launch. Some of the pre-launch thermal environments proved to vary significantly from the expected flight environments, and several heaters, heater controllers and power sources were diagnosed by ACAWS as possibly failed. Temperatures were well

below both the primary and secondary setpoints, and both primary and secondary heater switches were turned on, which were determined by the diagnostic reasoner as indicating that something was malfunctioning. None of the data signatures matched the model perfectly, however, and post-flight analysis confirmed that the heaters were behaving nominally but in an unanticipated environment. This yields a valuable requirement of a robust diagnostic system that the full range of system environments must be handled. Otherwise, "ops work-arounds" will be accumulated, requiring an ever more complex operations system and more training to differentiate between "real failure" and "unusual but nominal" situations.

Shortly prior to launch, when the ground umbilicals were disconnected, the data stream was reduced to a small trickle of critical data. The Orion communications system was shielded by the Launch Abort System until separation about 6 minutes into launch, so the only available data was through the Delta launch vehicle's payload interface until separation and the Orion spacecraft's data was transmitted to the MCC via the Tracking and Data Relay Satellite System. The transition from pre-launch to launch data had not been tested prior to flight. ACAWS handled the transition well. Any prior diagnoses remained, since there was no data that could exonerate any failures. Once data reconnected after Launch Abort System separation, some of the false diagnoses persisted due to an inadvertent test suppression that precluded the now nominal thermal data from exonerating the faults. A quick system reset cleared the situation and nominal diagnoses returned. Post-flight analysis led to a simple data adjustment and playback of the flight data resulted in correct diagnosis with no false positives or ambiguity. This points to another key requirement for a flight diagnostics and autonomy system, that when unanticipated flight data reveals untested conditions, the system must be quickly reconfigurable to adapt to the newly acquired information. Months of analysis, retesting and recertification of such systems will diminish their value significantly if not fatally.

A few other conditions were revealed during flight, but all were resolved within a few weeks, and subsequent playback of the complete mission was conducted with no false positives or missed detections during the flight, from Launch Abort System jettison to well past splashdown.

VIII. Conclusions

The technology is well beyond proof of concept but considerable maturation and scaling are needed to provide the objective evidence of the "proof of value" and "proof of readiness" needed for human space flight programs to build the ACAWS capabilities into program requirements and design baselines.

Programmatic infusion will require that programs either define new requirements not previously implementable, or determine that significant and measurable improvements can be made in the implementation of existing requirements, including lower cost, less risk, or improved performance. Performance improvements could include reduced processor or power needed for health state information management, or perhaps increased coverage of failures detected and managed on board, although that could be considered a new requirement. Programs will then need to see that the technology can be developed, tested and verified with acceptable cost, schedule and technical risks.

The project goal was to make as much headway toward these issues as possible, and to gain further clarity about the remaining issues so that follow-on phases and future technology projects can address the right issues. A potential risk for technology projects is that they continue to do proofs of concept, and raise Technology Readiness Level (TRL) from 4 to 5 over and over again.

The project achieved its scalability objective, using about 10% of the Orion downlink, covering all EFT-1 electrical power system components and most electrical equipment in vehicle subsystems. Although an estimate of the data required for complete coverage of fault management needs, as compared with data used only for operations, was not completed, the project showed that diagnostic systems can scale toward realistic operational requirements.

The project also achieved its objectives to operate in an operationally realistic setting, with opportunity to analyze variations between test and flight data, including environments, configurations, communications uncertainty and unanticipated data variability.

Future technology maturation that will help to prepare this technology for full-scale flight program infusion should include capabilities to:

- Continue to scale up sufficiently to analyze the performance metrics and system development effort that will be required of a fully operational system.
- Develop models and data across a more diverse set of subsystems, including power, data, and mechanical systems, to assure that fault detection and isolation methods integrate well.
- Improve and integrate the data filtering and levels of processing needed to distinguish between incorrect data and system faults, including sensor and data transmission faults.

- Develop more stringent methods for avoiding false positives and missed detections, and derive requirements for acceptable rates of incorrectness. Methods for developing and testing across the range of environment variability, system configurations, and data noise are a significant need for achieving accurate, reliable and trustworthy fault information.
- Design in methods to assure that new information can be quickly inserted into models and systems.
- Conduct model development cost assessment and analysis, including continued development of model automation from primary data sources. Cost and value uncertainty is a significant impediment to deployment of technologies that are promising but that have not fully proven their value where rigorous verification and certification is required.
- Provide analysis and determination of the requirements for the accuracy and trustworthiness of information, and the ability of fault management technology to meet the requirements.

The potential failure space of a complex system is vast, and systems excel at presenting new and unexpected ways to fail. Fault management technology designers must deal with the challenges of very large numbers of possible failures with a mature understanding of the fault management requirements and the programmatic, technical and cost risks remaining to be mitigated in order to deploy spacecraft with highly autonomous fault management capabilities. The Orion EFT-1 ACAWS project made significant strides toward demonstrating scaled up, accurate, trustworthy, and operationally significant fault information. Full-scale operational deployment of fault management technology faces significant remaining challenges that will need to be overcome to achieve difficult fault management autonomy requirements for future operations.

Acknowledgments

The authors thank Carlos Garcia-Galan of the Orion Program for his assistance with project formulation, advocacy and advice throughout the project. We thank the project engineering team at NASA ARC who assured successful performance during evaluations and flight: Jeremy Johnson for building and fixing the fault model; John Ossenfort and Adam Sweet for developing the diagnostics software; Vijay Baskaran and Silvano Colombano for designing, developing and testing the System Effects Reasoner; Lilly Spirkovska for leading the ACAWS User Interface group through many difficult display decisions; Irene Smith and Bill McDermott for developing, testing and retesting the ACAWS GUI; Rob McCann for planning the flight controller evaluation and designing test and evaluation scenarios The authors thank Jeremy Frank and Ann Patterson-Hine, NASA ARC and Richard McGinnis at NASA HQ for their project management and leadership without which the project would not have been possible. The authors thank the JSC Flight Operations Directorate for supporting the project from requirements definition and prioritization, through providing enthusiastic flight controllers to evaluate the system and provide invaluable advice for future work.

References

[1] Aaseng, G. B., Patterson-Hine, A., Garcia-Galan, C., "A Review of System Health State Determination Methods", 1st Space Exploration Conference, AIAA 2005-2528, Orlando, FL, 2005.

[2] M. Schwabacher, R. Martin, R. Waterman, R. Oostdyk, J. Ossenfort, and B. Matthews, "Ares I-X Ground Diagnostic Prototype", Proceedings of the AIAA InfoTech@Aerospace Conference, AIAA 2010-3354, Atlanta, GA, 2010.

[3] Ferrell, R., Lewis, M., Perotti, J., Oostdyk, R., Spirkovska, L., Hall, D. and Brown, B., "Usage of Fault Detection Isolation & Recovery (FDIR) in Constellation (CxP) Launch Operations", SpaceOps 2010 Conference, AIAA 2010-2181, Huntsville, AL, 2010

[4] Frank, L. Spirkovska, R. McCann, L. Wang, K. Pohlkamp, L. Morin, "Autonomous Mission Operations," 2013 IEEE Aerospace Conference, Big Sky, MT, 2013.

[5] Qualtech Systems Inc. 2015. Web site, http://teamqsi.com

6 Barszcz, E., Robinson, P., and Fulton, C., "Tools Supporting Development and Integration of TEAMS Diagnostic Models", Infotech@Aerospace 2011, AIAA 2011-1639, St. Louis, MO, 2011

7 Colombano, S., Spirkovska, L., Baskaran, V., Aaseng, G., McCann, R. S., Ossenfort, J., Smith, I., Iverson, D. L., Schwabacher, M., "A system for fault management and fault consequences analysis for NASA's Deep Space Habitat", AIAA SPACE 2013 Conference and Exposition, AIAA 2013-5319, San Diego, CA, 2013.

8 Deb, S., Pattipati, K. R., and Shrestha, R., "QSI's Integrated Diagnostics Toolset," AUTOTESTCON, 97, 1997 IEEE Autotestcon Proceedings, Anaheim, CA, 1997, pp. 408-421.

[9] Colombano, S., Spirkovska, L., Aaseng, G., Schwabacher, M., Baskaran, V., Ossenfort, J., and Smith, I.. "A System for Fault Management, Including Fault Consequence." 43rd International Conference on Environmental Systems (ICES). Reston, Virginia: AIAA, 2013.

[10] McCann, R. S., Spirkovska, L., Smith, I., "Putting Integrated Systems Health Management Capabilities to Work: Development of an Advanced Caution and Warning System for Next-Generation Crewed Spacecraft Missions", AIAA Infotech@Aerospace (I@A) Conference, AIAA 2013-4660, Boston, MA, 2013