NASA/TM-2013-216510

Advanced Caution and Warning System, Final Report — 2011

Lilly Spirkovska
Ames Research Center
Moffett Field, California

Gordon Aaseng
Ames Research Center
Moffett Field, California

David Iverson
Ames Research Center
Moffett Field, California

Robert S. McCann
Ames Research Center
Moffett Field, California

Peter Robinson
Ames Research Center
Moffett Field, California

Gary Dittemore
Johnson Space Center
Houston, Texas

Sotirios Liolios
Johnson Space Center
Houston, Texas

Vijay Baskaran
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Jeremy Johnson
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Charles Lee
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

John Ossenfort
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Mike Dalal
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Chuck Fry
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Larry Garner
Tietronix Software Inc.
Johnson Space Center
Houston, Texas

March 2013

NASA STI Program... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collects papers from scientific and technical conferences, synopsia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2013-216510

Advanced Caution and Warning System, Final Report — 2011

Lilly Spirkovska
Ames Research Center
Moffett Field, California

Gordon Aaseng
Ames Research Center
Moffett Field, California

David Iverson
Ames Research Center
Moffett Field, California

Robert S. McCann
Ames Research Center
Moffett Field, California

Peter Robinson
Ames Research Center
Moffett Field, California

Gary Dittemore
Johnson Space Center
Houston, Texas

Sotirios Liolios
Johnson Space Center
Houston, Texas

Vijay Baskaran
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Jeremy Johnson
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Charles Lee
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

John Ossenfort
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Mike Dalal
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Chuck Fry
Stinger Ghaffarian Technologies Inc.
Ames Research Center
Moffett Field, California

Larry Garner
Tietronix Software Inc.
Johnson Space Center
Houston, Texas

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California, 94035-1000

March 2013

Available from:

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

National Technical Information Service
5285 Port Royal Road
Springfield. VA 22161
703-487-4650

For the Enabling Technology Development and Demonstration Program,
Autonomous Systems and Avionics Project,
Integrated Systems Health Management Task –

Advanced Caution and Warning System (ACAWS)

Lilly Spirkovska, Gordon Aaseng, David Iverson, Robert S. McCann, Peter Robinson
NASA Ames Research Center

Gary Dittemore, Sotirios Liolios
NASA Johnson Space Center

Vijay Baskaran, Jeremy Johnson, Charles Lee and John Ossenfort
Stinger Ghaffarian Technologies Inc. at NASA Ames Research Center

Mike Dalal, Chuck Fry
Stinger Ghaffarian Technologies Inc. at NASA Ames Research Center

Larry Garner
Tietronix at NASA Johnson Space Center

Table of Contents

List of Figures

List of Tables

For the Enabling Technology Development and Demonstration Program,
Autonomous Systems and Avionics Project,
Integrated Systems Health Management Task –
Advanced Caution and Warning System (ACAWS)

Preamble

This is a living document. As the team learns about the problem(s) and corresponding solution(s), the document will be amended. We aim to maintain knowledge not only of what works well, but also of what did not work or did not pan out as expected. The reader will notice that the sentence tense changes from future to past as the document proceeds. This provides an account of what was planned versus what was accomplished.

1. Introduction

The Advanced Caution and Warning System (ACAWS) is a fault management tool that combines dynamic and interactive graphical representations of spacecraft systems, systems modeling, automated diagnostic analysis and root cause identification, system and mission impact assessment, procedure and flight rule (FR) identification, and interaction with other tools to help spacecraft operators (both flight controllers and crew) understand and respond to anomalies more effectively. Each of these capabilities provides critical support in monitoring the performance of vehicle systems as well as supporting the real-time decision process of MCC flight controllers and crew in connection with dealing with spacecraft anomalies and failures. In addition to real-time mission support, ACAWS' capability to create and interact with malfunction scenarios will support the analysis and training tasks associated with spacecraft operations.

The goals of the ACAWS project are:

- To develop the technologies to support vehicle operators as they plan for, train for, and fly a spacecraft mission.
- To develop an infrastructure that allows reuse and integration of multiple products, enabling the operator to focus on accomplishing mission tasks with minimal need of managing multiple software tools.
- To understand what the operators' needs are, what and how existing MCC tools can be integrated, what Integrated Vehicle Health Management (IVHM) technology can be used as is and what needs to be extended to meet the operators' needs, what is an effective concept of operations that incorporates IVHM technologies, etc. The product of the project is not just a prototype system, but the associated lessons learned in developing it. Although the project cannot necessarily drive a standard format for any future spacecraft program(s), the project can demonstrate the

benefits of specific formats and, more importantly, demonstrate the benefits of having standard data sets that can be reused across multiple projects of a program.

The focus of ACAWS is on the needs of both flight controllers and onboard crew. Although we expect flight controllers to continue to assist a crew in low-Earth orbit in dealing with system malfunctions, for future deep-space missions, the crew will need to accomplish some tasks autonomously due to communication time delays. We expect that providing similar tools – albeit perhaps with a different level of detail and different display formats or interaction methods – to the flight controllers and the crew could enable more effective and efficient collaboration as well as heightened situational awareness. In the remainder of this report, *operators* are used to refer to either flight controllers or crew.

The work described in this report is a continuation of the ACAWS work funded in fiscal year (FY) 2010 under the Exploration Technology Development Program (ETDP), Integrated Systems Health Management (ISHM) project. In FY 2010, we developed requirements for an ACAWS system and vetted the requirements with potential users via a concept demonstration system.

In FY 2011, we developed a working prototype of aspects of that concept, with placeholders for technologies to be fully developed in future phases of the project. The objective is to develop general capability to assist operators with system health monitoring and failure diagnosis. Moreover, ACAWS was integrated with the Discrete Controls (DC) task of the Autonomous Systems and Avionics (ASA) project. The primary objective of DC is to demonstrate an electronic and interactive procedure display environment and multiple levels of automation (automatic execution by computer, execution by computer if the operator consents, and manual execution by the operator).

The integrated team, known as System Health and Impact Evaluation for Decision-making (SHIELD), conducted a combined demonstration and evaluation of system health management on the Habitat Demonstration Unit (HDU) at the Desert Research and Technology Studies (D-RATS) in the Arizona high desert on August 23 – September 14, 2011. The integration of ACAWS and DC was demonstrated by ACAWS recommending malfunction (troubleshooting and recovery) procedures to the operator, who was then able to interact with those procedures using DC's procedure display. In a system without an adequate distribution of telemetered sensors to fully disambiguate the cause of failures, other forms of observations are needed to isolate to a proximate cause. To further disambiguate diagnoses, some of the recommended procedures served to guide the operator in providing these additional observations about the system to the diagnosis engine. We refer to this as *guided troubleshooting*. Other procedures were recommended to restore functionality. Through this partnership, SHIELD developed and demonstrated technologies needed for anomaly detection, fault diagnosis, and fault recovery.

We refer the reader to the FY 2010 ACAWS final report for information about the background, requirements, and general approach. In this report, we focus primarily on new developments. We begin by describing the integrated objectives of SHIELD. We do not discuss DC technology in much depth, but only describe aspects directly relevant to ACAWS.

2. Goal and Objectives

The goal of SHIELD is to mature ISHM and DC technologies that will enable deep space missions. These technologies, as stated previously, were demonstrated on the HDU as part of D-RATS 2011. The expected results of the experiment included the following:

1. Enhanced and new ISHM/DC capabilities developed, guided by DSH needs and constrained by DSH capabilities
2. User evaluations of the approach and information presentation concepts, leading to requirements vetting and generation of additional requirements
3. Lessons learned regarding capabilities maturation for anomaly detection, diagnosis, health management situation awareness presentations, procedure generation and interaction, and diagnosis-informed discrete controls.
4. Concepts of operation of ISHM/DC systems for deep space missions

The objectives of SHIELD address aspects of the well-known spacecraft flight control approach of FIW: Failure, Impact, and Workaround. The FIW aspect addressed by each objective is highlighted in the following list:

1. Mature anomaly detection (AD) capabilities [**F** I W]
    a. Develop a general capability (an AD executive) to use ARC-developed Inductive Monitoring System (IMS) with systems that transition through multiple operating phases, utilize multiple IMS knowledge bases (KBs) per operating phase, and continue monitoring even when some incoming data is invalid or explicitly bypassed by an operator
    b. Develop tools and methods to enable an operator (domain system expert, but not IMS expert) to develop and update KBs in situ. Requires developing methods/procedures for remotely guiding training process and tools to facilitate the process.
    c. Evaluate effectiveness of IMS for fleet supportability tasks
    d. Develop methods to present IMS results to operator & evaluate UI design and perceived benefits to operator situational awareness (SA)
2. Mature diagnostic capabilities [**F** I W]
    a. Develop a general architecture that utilizes Qualtech Systems Inc. (QSI) TEAMS-RDS for diagnostics
    b. Develop and evaluate effective (that is, SA-improving), usable, operationally beneficial and efficient methods to present diagnoses to operators
3. Mature information presentation and interaction methods that help an operator maintain SA of system health and system operations [**F I W**]; this is being accomplished via objectives #1-d and #2-b. Evaluate different levels of presentation: quick-glance, SA-focused iPad displays & extended analysis-capable console displays
4. Mature methods to assist operators with failure recovery [F I **W**]
    a. Develop and demonstrate a concept and methods for integrating diagnosis information with procedures

b. Expand procedure display capability to facilitate operator's interaction with procedure navigation and execution with varied levels of automation (LoA)
c. Develop tools and methods to facilitate resolution of malfunctions unforeseen by operations personnel, i.e., for a novel set of failures for which malfunction procedures have not been worked out in advance
    i. Determine how to assist operator with authoring procedures by utilizing diagnostic models and exploiting diagnostic tree (QSI's TEAMATE-like) functionality
    ii. Develop information presentation and interaction methods for presenting TEAMS/TEAMATE information to operator and attaining results of operator-performed system observations
d. Enhance procedure authoring & execution by proposing and evaluating new approaches for treatment of LoA

This is a rather extensive and ambitious set of objectives for a limited duration and DSH infrastructure that will continue to undergo its own development until its transport to AZ. We will develop the SHIELD system utilizing agile development methods, starting with core functionality and enhancing functionality as time allows. Some objectives will not be met.

3. Concept of Operations

SHIELD will run in multiple locations and will be getting either real-time data or delayed data, depending on location. A delay of 50 seconds each way is being artificially introduced for D-RATS 2011 to simulate and evaluate operations in deep space. SHIELD locations and the corresponding data update rate are shown in the following matrix:

Table 1: SHIELD operation locations.

Location	Data (& results) update rate
DSH	Real-time
On-site back room ("control tent")	Real-time ~~& delayed~~[1]
Remote MCC @ JSC ("MCC-H")	Delayed
~~Remote test support room (TBD) (JSC/N220)~~[2]	~~Delayed (TBD)~~

There will be three independent parts of SHIELD at D-RATS 2011: anomaly detection (AD), procedures, and integrated diagnosis plus procedures. Each of the parts will be integrated with the HDU and will be available to the crew in the DSH, controllers in MCC-H, and test support staff in the control tent.

[1] Not available, due to decision external to SHIELD.
[2] JSC decision was made to not have a remote test support team due to resources shortage, primarily personnel and equipment.

SHIELD/AD will run continuously and will be focused on monitoring the conditions in the subfloor area (where the computers are housed). SHIELD/AD will provide the deviation from nominal operating conditions, as learned automatically from archived data sets. As an evaluation of the effectiveness of SHIELD/AD for fleet supportability, AD will be trained on archived data before the DSH leaves JSC; its performance will then be monitored when set up in AZ using these same knowledge bases. Any deviations may reflect inconsistent assembly, transportation-induced problems, or just environmental differences. AD will then be retrained on nominal operating behavior in situ for a revised baseline.

SHIELD/Procedures subsystem will be available continuously and will run on-demand, with the operator selecting a procedure to display and execute from a table of contents. Procedure authoring will also be available on site (both in AZ and at JSC) for creation of new procedures or modification of existing procedures. SHIELD/Procedures will provide an electronic and interactive procedure display environment and multiple levels of automation (automatic execution by computer, execution by computer if the operator consents, and manual execution by the operator).

SHIELD/Diagnosis+procedures subsystem will run as a demonstration, primarily for concept evaluation purposes. A subset of the power system will be modeled and multiple (independent and not concurrent) failures will be introduced (seeded with minimal interference to other on-going experiments). An operator will interact with SHIELD/Diagnosis+procedures to determine the cause of failure, execute troubleshooting procedures to disambiguate a diagnosis if necessary, and execute recovery procedures to restore HDU system functionality. Controllers in MCC-H will be able to follow along but with a 50 seconds delay. Following each scenario, operators (crew and controllers) will be interviewed for opinions and improvement suggestions.

All data and results will be archived for post-experiment analysis.

4. Architecture

SHIELD is a system of systems, as shown in Figure 1. It is being developed to provide general capability, with minimal code specific to the DSH or the demonstration subsystem selected. For the DSH demonstration, SHIELD will run on Linux laptops; the SA displays will also run on Apple iPad tablet computers. Communication of data between systems will utilize ICE (Internet Communication Engine), an object-oriented middleware.

The ISHM/ACAWS portion consists of COTS tool QSI's TEAMS-RDS, NASA/ARC tool IMS, NASA's JMEWS[3] software, ACAWS graphical user interface (GUI), and supporting

[3] JMEWS – Java Mission Evaluation Workstation System – is a tool prepared for NASA/JSC and used in the ISS MER (Mission Evaluation Room, the backroom support for ISS mission controllers).

infrastructure code. Development is in Java and C++ and will run on Linux OS, though it will be portable to various operating systems.

The DC portion consists of the Procedure Display (PD), the Universal Executive (UE), and the Level of Automation (LoA) server.

A simulator of the DSH, to be developed by the HDU team to assist with testing DSH/Avionics, was to be used for testing SHIELD. Simulator fidelity was to vary depending on the DSH subsystem being simulated, with some fairly accurate simulations and others non-existent. Unfortunately, the simulator was not developed as planned. As stated below, we tested, as much as possible, with an alternate approach that used playback of logged HDU operations.

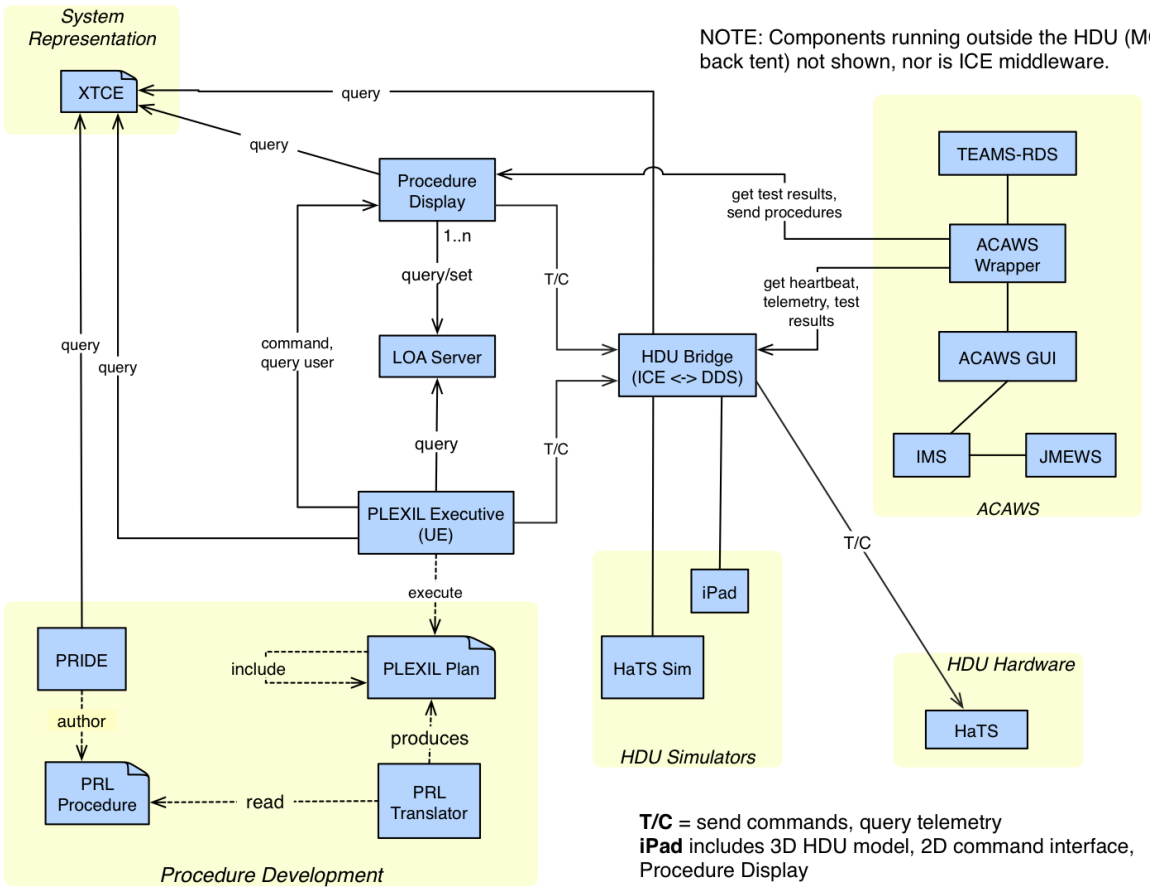

Figure 1: SHIELD system diagram.

5. Tasks and Schedule

The following lists provide a high-level description of the tasks to be accomplished, with the enumeration biased toward the ACAWS tasks, with DC tasks listed only when directly relevant to ACAWS.

Task #1. Anomaly detection executive development
Task #2. AD knowledge base development
Task #3. Diagnosis executive development
Task #4. Diagnostic model development
Task #5. Guided troubleshooting development
Task #6. AD GUI development
Task #7. ACAWS GUI development
Task #8. iPad GUI development
Task #9. PD GUI enhancement
Task #10. UE/LoA enhancement and updates for new PRIDE version
Task #11. Communication middleware refactoring

System integration

Task #1. Procedure recommendation handshake between ACAWS & PD
Task #2. Guided troubleshooting handshake between ACAWS & PD
Task #3. AD/ACAWS/PD GUI integration with DSH (via ICE middleware)
Task #4. AD/Diagnosis executive integration with DSH
Task #5. Test with DSH hardware in the loop

Field Preparation

Task #1. Diagnosis demonstration scenario development
Task #2. Prepare computing resources for field operations
Task #3. Train AD on nominal operating behavior at JSC
Task #4. Training curriculum development
Task #5. Operator training
Task #6. Evaluation approach and questionnaires development
Task #7. Demonstration scenario procedure development
Task #8. Get SHIELD experiment on D-RATS timeline

Field Experiment

Task #1. Test with DSH hardware in the field
Task #2. Update AD knowledge bases, as required
Task #3. Conduct demonstrations & evaluations

Wrap Up

Task #1. Data analysis
Task #2. Final report

A general schedule for SHIELD is provided in the following table:

Table 2: SHIELD schedule

Task	Time period
Software Development	Oct. 2010 – June 2011
System Integration	July 2011

Field Preparation	July 2011 – Aug. 15, 2011
Field Experiment	Aug. 23 – Sept. 14
Wrap Up	Sept. 14 – Sept. 30

6. SHIELD/Anomaly Detection

Investigative Domain

We developed a general capability for anomaly detection (AD) using the Inductive Monitoring System, IMS. An AD executive system manages connection to the communication middleware (ICE) to retrieve the real-time telemetry, selects the appropriate knowledge base (KB) for a given operating phase, marshals the data into the appropriate vector, sends it to IMS, and posts the IMS deviation scores for the composite vector and each parameter individually back to ICE. Through ACAWS, we developed an extended capability to enable the AD executive to transition through multiple operating phases, utilize multiple IMS knowledge bases (KBs) per operating phase, and continue monitoring even when some incoming data is invalid or explicitly bypassed by an operator.

The AD system was demonstrated at D-RATS by monitoring the conditions in the HDU subfloor area. The monitoring continued throughout the field experiment and displays were available to the HDU controllers (in the control tent) and to the operator during in-the-HDU demonstrations.

The subfloor has 24 sensor values (from 21 sensors) sent through either the wireless network system or through a wired connection. AD is agnostic to transmission path because all telemetry goes through ICE. With guidance from a subsystem expert, we selected 20 sensor values to monitor, as follows:

- 16 thermocouple sensors sensing temperatures throughout the subfloor area (2 per segment with the HDU logically separated into 8 segments)
- A humidity sensor that senses both humidity and temperature
- An airflow sensor that senses both airflow and temperature

Four sensor values were omitted, as follows, because they are not sensing the overall environment in the subfloor:

- An airflow sensor, sensing airflow and temperature of the airlock
- 2 thermocouples sensing temperatures on a specific electronics box.

The humidity sensor, providing both humidity and temperature readings, was omitted once we arrived in the field. The sensor was initially associated with an RIU that was mistakenly omitted from the HDU avionics software build. After a few days of troubleshooting, the HDU team decided to re-associate that sensor with a different RIU. IMS requires a full vector of values for training data so the choice was to either disregard all otherwise-valid data already captured or disregard that sensor. We chose the latter option primarily because we had already exceeded approximately 30% of our field test duration.

In anticipation of possibly different nominal operating behavior during day or night operations, we added a computed parameter for time since midnight. Thus, the training

vector was composed of the instantaneous values of each monitored parameter and the time of day.

Training data for the 2011 HDU was not available during development of ACAWS. We therefore relied on data collected from the 2010 HDU. Three days of nominal operating behavior was available. This would be adequate to capture operating behavior only if there is very little variability between different operating days or conditions. Still, it provided an opportunity to begin the process of developing a KB with the expectation that further tuning would occur using 2011 HDU data, both from integrated tests while the HDU was still at JSC and from data captured once the hardware/software was transported to and installed in the field. We also had access of three days of off-nominal operating behavior that we could use for testing the knowledge base.

It was hoped that additional training data would be available during integration testing at JSC. HDU hardware and avionics integration problems resulted in a very short duration of nominal HDU operations. We were therefore unable to compare the operating behavior of the 2010 HDU configuration to the 2011 configuration, eliminating an ACAWS objective of evaluating the effectiveness of IMS for fleet supportability tasks.

User Interface

IMS deviation scores are presented to the operator on three display formats, each with a different level of detail, as shown in Figure 2. The sensor data is also available to the operator in an integrated display format, providing both tabular display and graph display, as shown in Figure 3. The display formats are developed using the JMEWS software developed for JSC and used by controllers in the support rooms for ISS operations.

Figure 2: Anomaly detection display formats. Upper left display shows a meter for the instantaneous value of the composite deviation score. Upper right display shows meters for the instantaneous values of each of the parameters being monitored. The meters are arranged to parallel the clockwise arrangement of HDU segments A through H with two temperatures per segment. We also added a time-of-day parameter to accommodate possibly different behavior during day and night operations. The bottom display graphically shows deviation scores over time arranged as in the meters display.

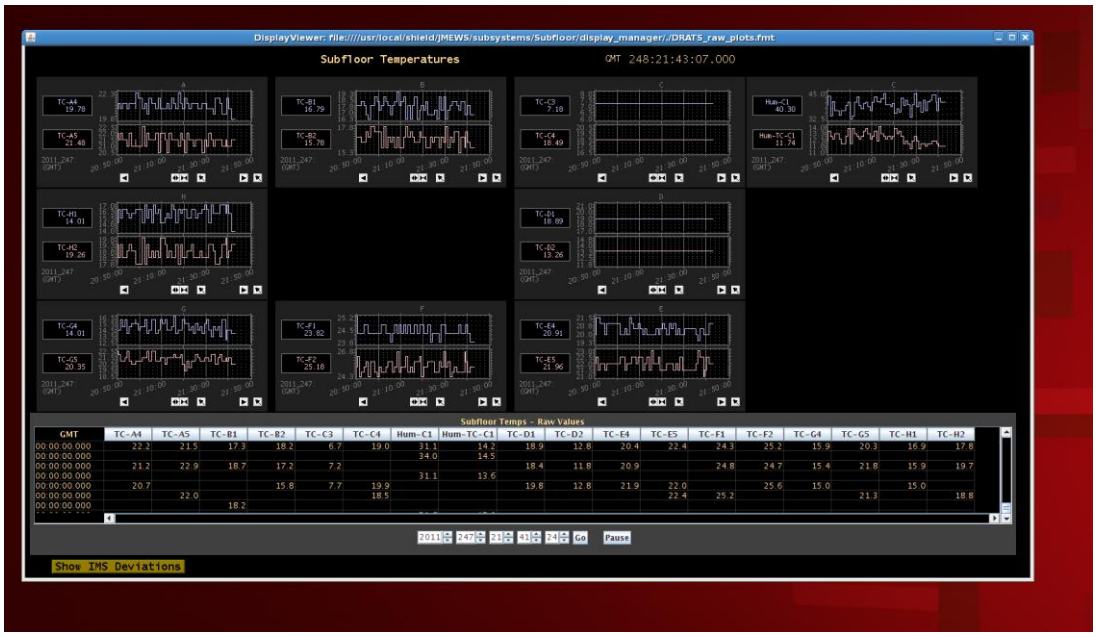

Figure 3: This display shows the subfloor temperatures in graphic and tabular formats.

Results

The AD-related goal for SHIELD, as stated in Section 2, was to mature anomaly detection by developing a general AD executive for IMS, developing tools and methods to enable an operator to develop and update KBs in situ, evaluating the effectiveness of IMS for fleet supportability tasks, developing methods to present IMS results to an operator, and evaluating the UI design and perceived benefits to operator situational awareness (SA). As anticipated (see Section 2), we did not accomplish each of these tasks. Nevertheless, we made big strides toward them.

We developed a general AD executive that facilitates use of IMS with systems that transition through multiple operating phases, utilize multiple IMS knowledge bases (KBs) per operating phase, and continue monitoring even when some incoming data is invalid or explicitly bypassed by an operator. Because these capabilities were not necessary for the HDU, they were instead verified on a Space Shuttle LH2 fuel loading operation demonstration in April 2011.

To evaluate the effectiveness of IMS for fleet supportability, we planned to train IMS on nominal operations data while at JSC and then compare HDU operating behavior in the field. Unfortunately, due to HDU hardware/software integration issues prior to transport to the field, the final HDU configuration was reached a few days after its arrival in the field. Since the before-transport configuration differed from the in-the-field configuration, a comparison with IMS would have been meaningless. Thus, it was not possible to meet this objective.

Once HDU configuration was stabilized, we determined that subfloor temperature nominal behavior is very steady and, thus, not much nominal data is needed to train IMS. The temperature sensors were not calibrated, resulting in readings from two adjacent sensors differing by ~ 15 deg C. This was not a factor for IMS since it looks only for differences in nominal behavior rather than specific values.

We detected the first failure – a WSN failure -- with IMS on the first day after we trained it on a few hours of the previous day's nominal data. We continued to detect subsequent (and frequent) WSN failures.

IMS also detected rising temperatures (at one point, it was 4 degrees in 15 minutes) in the computer area in the afternoons. The HDU team decided that should be considered nominal.

Each day, IMS was retrained, incrementally adding the previous day's nominal data to the training set. Because no operators were available to train on how to do this, the team did the training. Still, the process was refined; we expect training non-experts to perform the operation will not be too challenging.

Because of its usefulness in detecting WSN failures, the field controllers found AD valuable as AD was typically the first to determine that a failure had occurred. AD was also demonstrated to the operator/evaluators. The results of the evaluations are discussed in the *User Opinion* section below.

7. SHIELD/Diagnosis+Procedures

Investigative Domain

Developing a diagnostic model for the full HDU is out of scope for this project. To select a reasonable subset of the system for demonstration purposes, we considered a number of characteristics for a reasonable investigative domain. First, the diagnosis of selected failure scenarios needs to be complex, both in the failure and the annunciation of the failure. Some desirable characteristics of such a failure include the following:

- Requires substantial system knowledge to perform manually
- Has a failure signature that looks similar to the signature of other possible failures, thereby providing a potential ambiguity of which failure happened
- Requires analysis to determine the failure, but is not so complex that it is difficult to explain in a demonstration presentation

Second, the selected subsystem needs to have enough sensors to allow for automated diagnosis. We wanted at least three scenarios to demonstrate different aspects of the technology, as specified below:

1. An unambiguous diagnosis for which a recovery procedure would restore functionality.

2. An ambiguous diagnosis with a small group of possible failures that are explained by the telemetered data. This would lead to a malfunction procedure that would require operator observations to be input to TEAMS to further disambiguate to the actual (seeded) failure. At that point, a recovery-type malfunction procedure could be invoked.
3. An ambiguous diagnosis that resembled an "unanticipated" failure. The ambiguity group, that is, the group of potential diagnoses that explain the failure signature, would be disambiguated via guided troubleshooting, with TEAMS suggesting the next-best observation for the operator to perform until either the diagnosis becomes unambiguous or there are no more operator observables that would help with disambiguation.

Since this year's HDU field tests (other than SHIELD) focused on nominal operations, each of the recovery and troubleshooting procedures would be generated by the SHIELD project.

The last consideration for investigative domain selection was criticality of the (seeded) failure. To be part of the integrated operations test, the given ground rule from HDU was that a seeded SHIELD failure would not cause field exercise work stoppage; any non-SHIELD planned operations must continue during a SHIELD demonstration. Further, we wanted a failure that requires immediate resolution. This would ensure that the crew would exercise ACAWS in an on-board capacity rather than deferring the problem to an MCC controller.

With these objectives in mind, we selected a subset of the power subsystem, specifically a failure of the power distribution unit (PDU) that provides power to the wireless sensor nodes (WSN) and solid-state light modules (SSLM; the overhead lights distributed throughout the HDU). The connectivity of PDU B/1 to other components is shown in Figure 4.

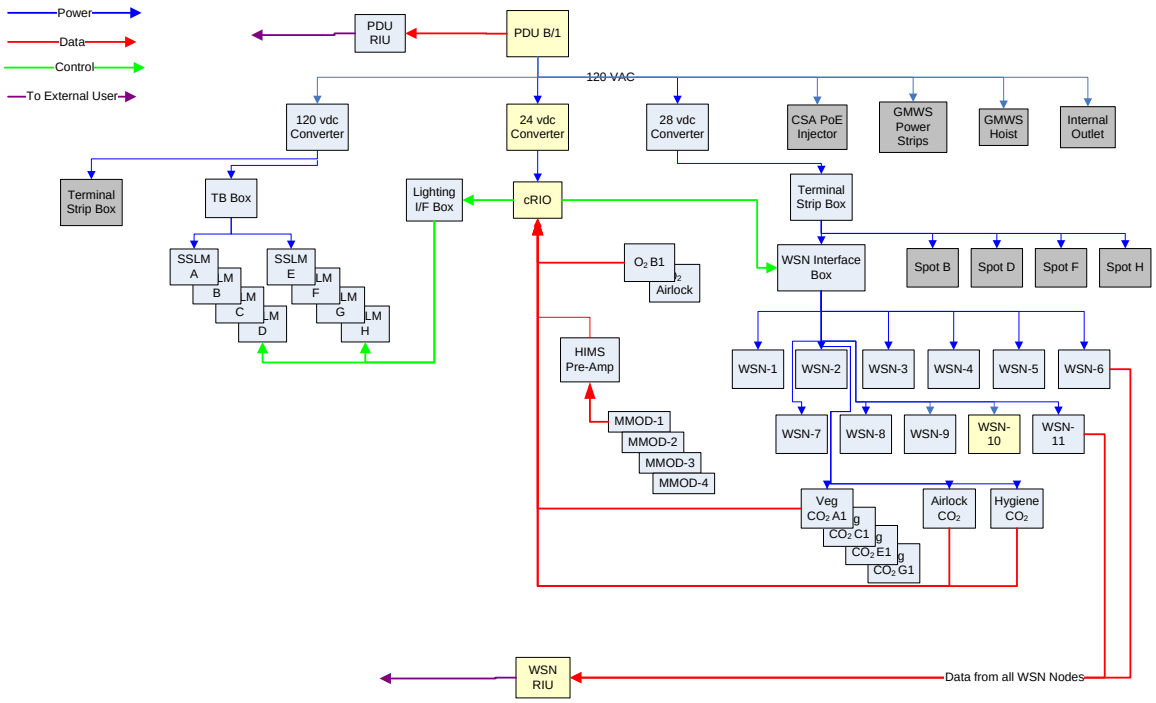

Figure 4: PDU B/1 power string.

A number of failures in this power string have similar signatures, leading to complexity in the failure and annunciation of the failure. For instance, it is not immediately obvious whether a failure is in the PDU itself, the 24 VDC converter, the 28 VDC converter, the cRIO, a single WSN, or an RIU (remote interface unit; software that reads and publishes WSN data). Further complexity arises because loss of power results in loss of data that is needed to determine system state. It requires both sensor data and operator input to diagnose failures. It includes both hardware failures and software failures, and the ability to differentiate between the two. For example, an RIU crash results in loss of data that could look much like a power or avionics component failure. Finally, the failures are each very significant but not catastrophic, and their impacts vary from fairly isolated to fairly broad.

Demonstration Scenarios

Five seeded failures were targeted for the demonstrations, addressing the objectives for three scenarios described earlier. In the simplest failure, a single WSN will be commanded to OFF. This will result in sensor data loss for up to 10 sensors. Not all sensors transmit at the same rate; thus, effects of each sensor drop out will be felt over a period of time. Depending on which WSN is selected, the failure may look like an RIU failure because both failures would show data drop out on several sensors. TEAMS will disambiguate between them by using RIU heartbeat data – present whenever an RIU is running. Impacts of a WSN failure are limited to loss of data on that node.

For the second demonstration failure – ambiguous diagnosis with anticipated malfunction procedure, the 28 VDC converter will be commanded to OFF. This will cut the power to all WSN motes and some of the CO2 sensors. Sensor losses would appear over time (up to one minute) as the sensors are transmitted at various rates and not tightly synchronized between nodes. Also, software tests for data drop out require multiple misses to be

verified, so it could be a few minutes before a diagnosis stabilizes. The malfunction procedure will direct the operator to check the exterior spotlights, if able, to disambiguate between the two most likely suspects of 28 VDC converter and cRIO 9477 Control Card failure.

For the third demonstration failure – ambiguous diagnosis with unanticipated malfunction procedure, the 24 VDC converter will be commanded OFF. As for the 28 VDC converter failure, many sensor losses would be annunciated over a period of a few minutes. The ambiguity for the cause of the failure is between the 24 VDC converter, a cRIO failure, and the 28 VDC converter. At this point, TEAMS will offer guided troubleshooting steps. Each step aims to most efficiently disambiguate between the failure suspect remaining in the ambiguity group. It selects the next best step ("test" in TEAMS parlance), using criteria such as most critical item to exonerate first, least expensive observation to perform, and availability of equipment to perform observation. TEAMS will initially disambiguate between the 28 VDC converter and the other two based on the exterior spotlights (as for the second demonstration). Once that is exonerated, disambiguating between the other two candidates will require going in the subfloor area.

As stated for the third demonstration failure, a 24 VDC converter failure and cRIO failure have the same signature when considering only automated observations. To disambiguate between them, a manual observation is required. In particular, the operator must determine whether the 24VDC converter indicator light is ON. If so, the converter is operational and hence, the cRIO must be failed to explain the observations. If the indicator light is OFF, the converter has failed. The system impact from either failure is similar – the cRIO (either because of its own failure or the failure of the upstream power supply) control function that commands 28 VDC power switches closed is lost, resulting in loss of data on all nodes. Additionally, if the failure is the power supply, the cRIO is impacted.

Finally, the fifth demonstration failure that can be used to illustrate ambiguous diagnosis with unanticipated malfunction procedure is a failure of one of the cRIO's internal cards, specifically the power control card. The signature of this failure based on automated observations looks similar to a 28VDC converter failure in that all of the WSNs will lose power. The 24 VDC and cRIO are exonerated by 2 O_2 sensors that are read by the cRIO but powered by a different power circuit. To disambiguate, the operator can turn on one of the spotlights that is powered by the 28VDC converter. If the spotlight is ON, the 28VDC converter is exonerated. If the spotlight is OFF, the cRIO power card is considered the failure.

Demonstration Model

The determination of diagnostic state for the ACAWS system was accomplished using QSI's TEAMS-RT runtime diagnostic engine that relies upon the use of a dependency matrix (D-matrix) compiled from a user-generated, domain-specific TEAMS Designer model. TEAMS-RT is able to query that D-matrix in near real-time in order to determine the most likely candidates for a given set of failures. The TEAMS Designer model (see Figure 5) was modified from an earlier TEAMS model developed by NASA/JSC contractor Aaron Schram for the 2010 D-RATS campaign. The TEAMS model is developed from the schematics of the

HDU and is partitioned hierarchically into seven subsystems: Power, Avionics, Communications, Geolab, Environmental Control and Monitoring, Crew Accommodations and Food Production. A breakdown of the major subsystems can be found in Appendix D.

The TEAMS model was modified from the 2010 D-RATS campaign to support the failures scenarios identified by the SHIELD team. Updates included:

- Additional ports and failure modes for Power Distribution Unit (PDU) B-1
- Added PDU B-1 electrical current (amperage) tests
- Renamed and added components along power fault propagation paths to reflect the current architecture
- Added various manual tests that were required to disambiguate failures
- Changed TEAMS "functions" to support accurate D-matrix generation
- Added cRIO and interfacing components along with failure modes and appropriate tests to reflect the current architecture
- Added 82 sensors that are connected to the Wireless Sensor Nodes (WSN)
- Updated the WSNs and added the corresponding tests
- Modeled various components that provided crew accommodation
- Updated the avionics fault propagation paths to reflect current configuration

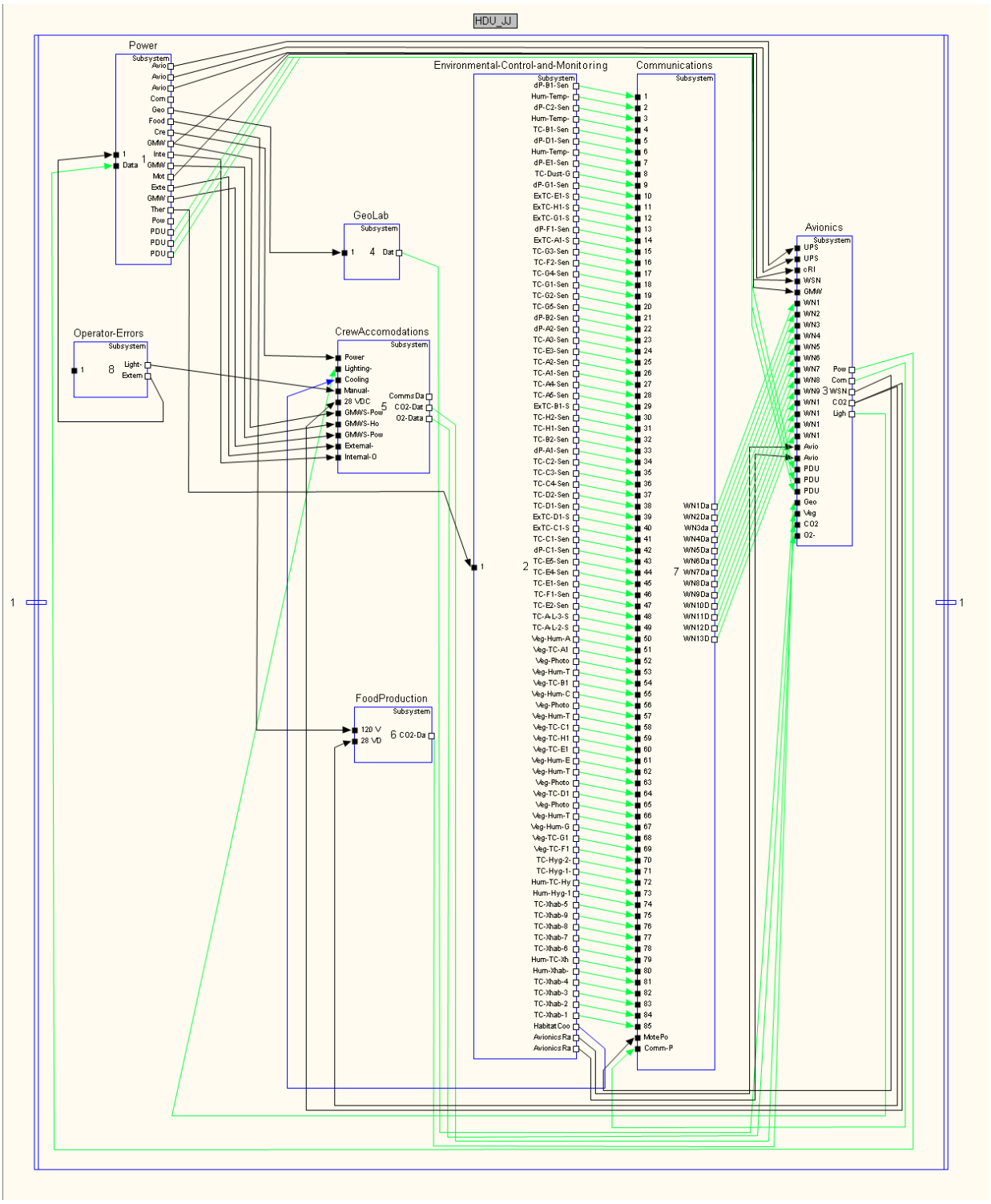

Figure 5: Demonstration TEAMS model.

Demonstration Diagnostic Model and Executive

The diagnostic executive portion of ACAWS is written in Java using Java Native Interface (JNI) to query TEAMS-RT utilizing the TEAMS-RT API. This "wrapper code" consists of both an input side to feed TEAMS-RT and an output side to take the returned TEAMS diagnosis and present it to the user in a useful way. On the input side, ACAWS takes outputs from the HDU system and translates them into pass/fail test results for TEAMS-RT to process. This interface needs to talk to the ICE middleware in order to retrieve the system outputs and also must talk to the TEAMS-RDS server in order to pass along the translated results. There are 4 main types of output coming from the HDU over the ICE middleware that need to be translated and passed to TEAMS:

1. Pass/fail test results based on telemetered-sensor observations – In the HDU, elements called interfaces are responsible for publishing out telemetry values from the sensors. These interfaces can be either Remote Interface Units (RIUs) or system controllers, with at least one controller being active for a given subsystem at any one time. Sensor values collected at various points in the system are transmitted by these interfaces over the ICE middleware. For the purpose of the ACAWS GUI, those sensor values can be displayed as-is to the user to give insight into the operating conditions of the HDU. For TEAMS-RT however, the sensor values must be compared with upper- or lower-bound limits or expected discrete state values and evaluated to either a pass or a fail result. These evaluations are performed at the subsystem level by the HDU system controller code and sent out over the ICE middleware, much like the telemetry data itself. This test result set can then be read by the ACAWS diagnostic executive and passed into TEAMS-RT via the TEAMS-RDS API.

2. Telemetered-validity-based observations – Along with the sensor values being sent out by each interface over ICE, a validity flag for that sensor is sent in the same packet to denote whether the sensor value can be trusted. If, for instance, a WSN mote fails and all of the sensors attached to that mote go dead (i.e. - stop returning current values), the normal behavior of the RIU is to continue publishing the last known value for those sensors. To distinguish between a stale value that might represent a failed component and a 'good' value, the validity flag for that sensor value must be checked along with the value itself. Test points were added to the TEAMS model to represent these validity checks. If the flag is valid then the appropriate test is passed, otherwise the test is failed.

3. Telemetered-heartbeat observations – The software running on each of the HDU interfaces is responsible for sending out a heartbeat message to the ICE middleware to indicate that it is healthy. In the case where an interface goes down due to software or hardware failure, a missing heartbeat message will alert the system that the interface is no longer working properly. The heartbeat message topic on ICE is subscribed to by ACAWS and associated with a watchdog timer. If the heartbeat message for any interface is not received in its expected time window, the associated test is failed in TEAMS-RT.

4. Procedure-driven observations - In addition to input sources from the HDU software, ACAWS also receives test results from the Procedure Display. As already discussed, the telemetered sensor observations are not sufficient for disambiguating most failures and, thus, operator-performed ("manual") observations must be provided. This is accomplished via procedures, utilizing DC's Procedure Display (PD). Each procedure is associated with manual tests in the TEAMS model. As the procedures are performed, the test results – provided by the operator – are sent to ICE. In much the same way that test results based on telemetry are passed in, the PD-based results must be mapped in the ACAWS executive to the proper TEAMS index and passed into TEAMS-RDS. For more information on the procedures see Appendix C: Diagnostic and Repair Procedures.

All of the test results above are subscribed to via ICE middleware and added to test result arrays for TEAMS-RT processing. There are several implementation aspects involved in formatting and buffering those arrays before passing them to TEAMS, however. Much of this work was done because of timing issues in the HDU telemetry stream. In particular, the lack of framing and synchronization in the HDU data posed some challenges when designing the diagnostic executive code. A description of the implementation and reasons behind it are summarized below:

1. Buffering Pass/Fail/Unknown test results – TEAMS standard diagnosis (resulting in a list of "bad", "suspect", and "unknown" components) is computed based on a history of past test results. This alleviates issues with transients, where a single out-of-limit sensor reading, for example, could result in the assumption of the existence of a fault. TEAMS standard diagnosis uses a "wait and see" approach before concluding a fault exists and providing a diagnosis of the alleged fault. Unlike the standard diagnosis, however, TEAMS minimum diagnosis (resulting in a list of "minimal" and "residual" components) generates a new diagnosis for each set of data depending only on the data received at that iteration. Due to the distributed nature of the telemetry in which each HDU controller is responsible for publishing a subset of the sensor values in the system, a test result vector composed only of the most recent data received from any one controller is not sufficient to capture the entire current state of the system. The solution is to buffer the last known state of each test – whether pass, fail, or explicitly computed as "unknown" – on the executive side and pass a complete test result vector into TEAMS-RT at every iteration.

2. Windowing of test result arrays – Another issue caused by the telemetry being published in subsets from each controller is that there are many more result sets being published than TEAMS-RDS can process in real time. The fastest rate at which TEAMS-RDS can return a diagnosis is approximately once per second. The ACAWS wrapper was processing on average around 15 messages per second from the HDU, of which anywhere from 2 to 6 of those were evaluated to Pass/Fail arrays for TEAMS-RT consumption. Compounding this issue is the fact that the ICE publish/subscribe middleware can sometimes be 'bursty' - not sending packets for a short period of time and then flooding the system in the span of less than a second in

an effort to catch up. During periods when the diagnosis is frequently changing, these factors can back up the processing and result in the diagnosis output falling out of synch with the current state of the system. The solution to this problem is another form of buffering, called windowing, wherein multiple test result arrays are combined and passed to TEAMS-RDS as a single array if they are received within a user-specified time-window. For the purpose of this demonstration we experimented with both 1-second and 2-second windows of data.

As mentioned earlier, the ACAWS executive also subscribes to test results returned by the PD during execution of procedures recommended by ACAWS. The procedures generally fall into three categories: single-test troubleshooting procedures; multi-test, pre-generated troubleshooting (diagnostic or malfunction) procedures; and function recovery (repair) procedures.

Troubleshooting procedures are recommended when TEAMS-RT is unable to diagnose the failure down to an unambiguous cause. Two forms of troubleshooting procedures were generated for the demonstration. We implemented two approaches for recommending single-test troubleshooting procedures: rule-based and model-based.

but has determined that a manual test could be performed that may help to reduce the ambiguity set further. If the ambiguity is reduced to the point of determining a single probable cause then the procedure recommendation will try to find a recovery procedure to either repair or replace that failed component. The rule-based solution was generated by ACAWS team members based on accumulated expertise with the failure scenarios and with the HDU hardware. The model-based solution relies on the TEAMATE interface within TEAMS-RDS. By analyzing the current health state of the system components as reported by TEAMS-RT, TEAMATE is able to determine the next best test to perform in order to most efficiently disambiguate the failure. Some progress was made in integrating the TEAMATE functionality for this demonstration, but additional work is needed to understand its capability and have its recommendations reflect a domain expert's recommendations. See the "lessons learned" section for more information.

In regards to the output side of the ACAWS code, much of the information presentation was performed by the ACAWS user display discussed earlier. The separation of ACAWS GUI and ACAWS diagnostic execution was a design decision intended to modularize functionality. This division of functionality also allowed us to run multiple ACAWS displays while subscribing to the same diagnosis output, giving a singular, coherent view of the system to multiple operators. Both the minimal and standard diagnoses as well as the procedure recommendations were published to ICE from the ACAWS wrapper for consumption by the user interface.

Demonstration Procedures

As described in the previous sub-section, the demonstration scenarios require recovery and troubleshooting procedures. These procedures are written in the Procedure Representation Language (PRL) and authored in the Procedure Integrated Development Environment (PRIDE). The Procedure Display (PD) is the graphical user interface for

viewing, selecting, and executing procedures. A detail hidden from the user, procedure execution is achieved by automatically translating PRL procedures into the Plan Execution Interchange Language (PLEXIL); this PLEXIL representation is executed by the Universal Executive (UE). Level of Automation (LOA) for each procedure step and instruction is managed by a separate entity called the LOA server. PRL, PLEXIL, PRIDE, PD, UE, and LOA server, all components of the Discrete Controls project, are integrated to provide a capable procedure authoring and execution environment to the operator. ACAWS interacts only with the PD.

ACAWS recommends procedures to the operator by posting an ICE middleware message to which the PD subscribes. If procedures contain steps that require operator observations (so-called "manual tests"), the PD presents the instruction to the operator and then provides the operator's answer back to the ICE middleware for consumption by the diagnosis engine as just another test result. We make no distinction between sensor-based test results and operator-observation test results. Figure 6 illustrates the communication between the various SHIELD software and HDU software both in the HDU and at MCC.

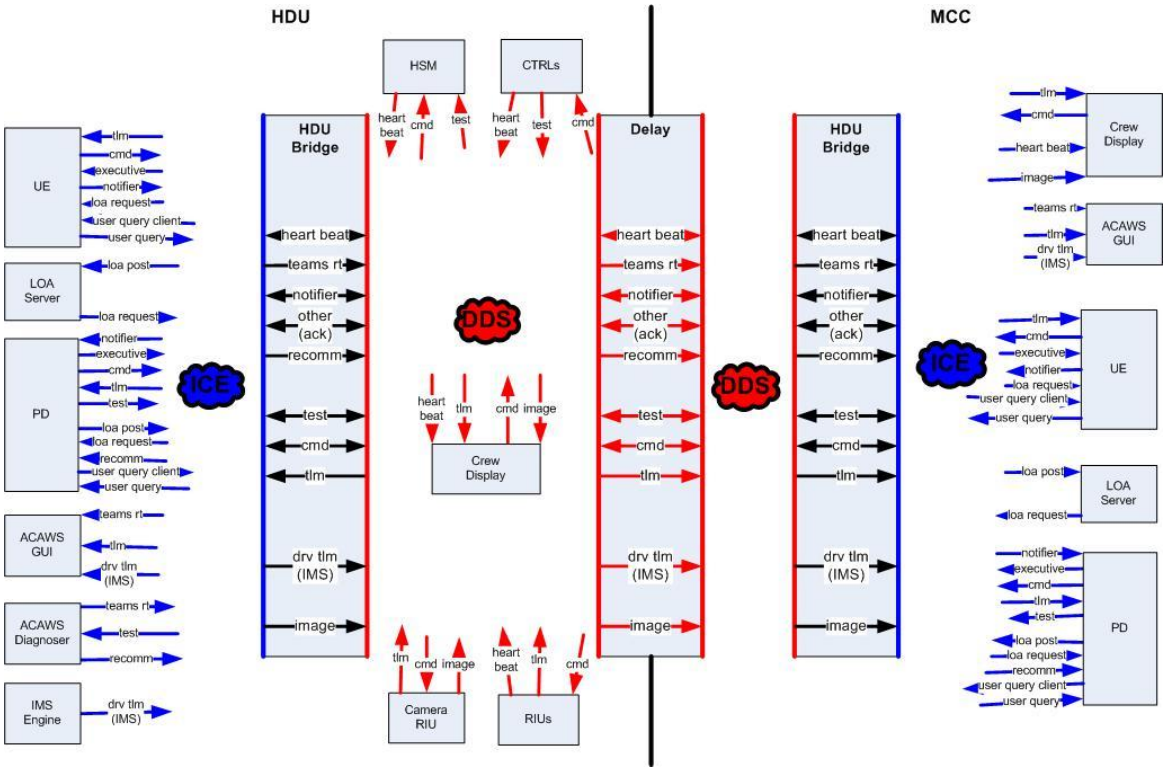

Figure 6: HDU and SHIELD software communication architecture.

The malfunction procedures, prior to being represented in PRL, are shown in Appendix C.

User Interface

The ACAWS user interface for this fiscal year builds upon the UI concept demonstrated last fiscal year. The general framework for the interface is shown in Figure 7. One of the key objectives for the framework is providing flexibility to support the operator to work how she/he wants rather than dictating a certain approach. Each of the panels (e.g., diagram,

system health annunciators, C/W msgs, etc.) is independent yet interaction with each panel is coordinated with the other panels, that is, a diagnosis component selected in the diagnosis panel also selects the components in that diagnosis on the diagram. A panel can be resized; moved to a different location within the window; "torn" off the main window into its own window and placed on the same display monitor or an adjacent monitor; hidden; or duplicated to contain another system health annunciator group, for instance. Multiple configurations of panels can be saved, allowing each operator to set up the panels as desired for different tasks. For example, an operator may have one panel configuration for monitoring that deemphasizes failure impact, procedures, and flight rules, a different configuration for analysis that focuses (and assigns more display real estate) on those panels. The goal for this year's work with regard to the UI framework was to develop the general capability in Java, establishing the path for future integration with other MCC tools under development, such as MCT, or existing tools, such as MSK and RT-Plot.

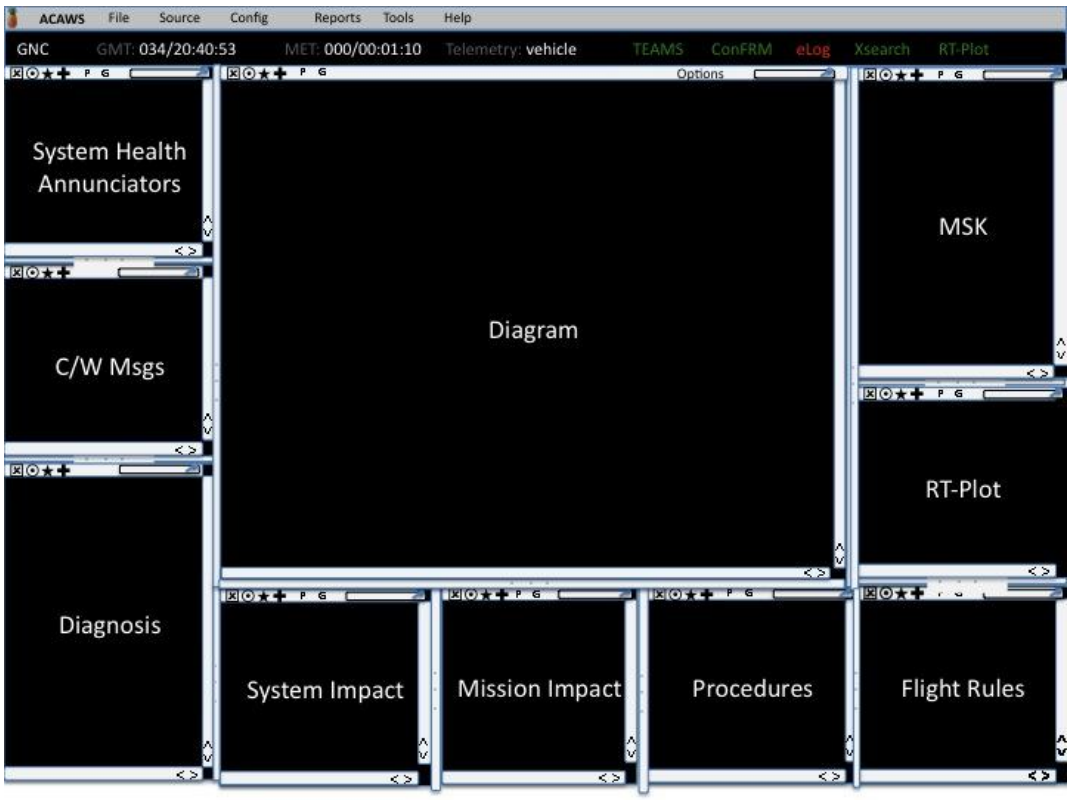

Figure 7: ACAWS (generic) user interface framework.

Specific display formats that reside within the ACAWS framework were developed for the demonstration scenarios. A system health annunciator arrangement was specified by the MOD customer and is shown in Figure 8.

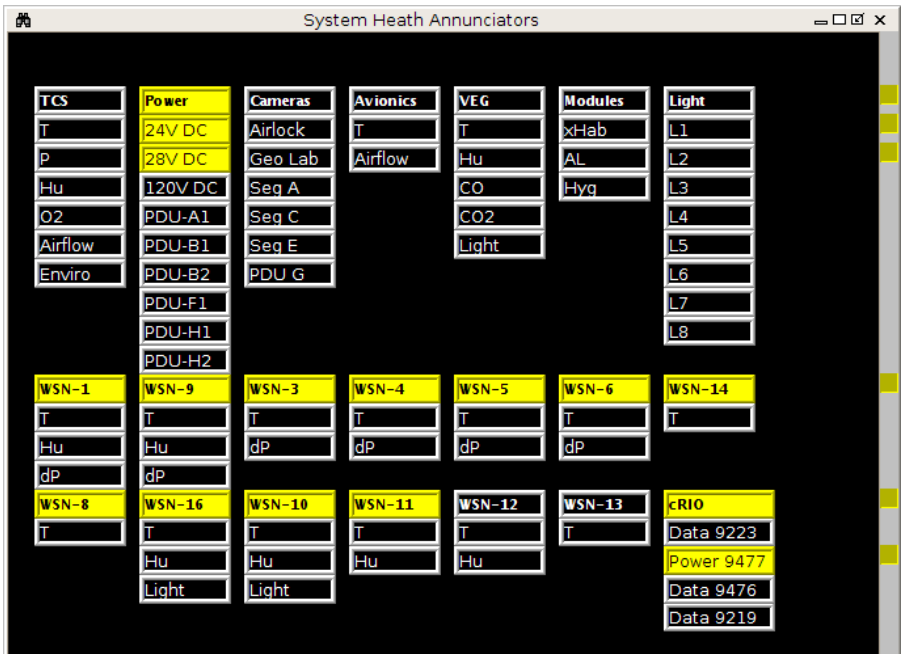

Figure 8: HDU system health annunciators.

Diagnoses from TEAMS are being displayed in a tree format rather than long strings of names that are representative of the diagnostic model but do not necessarily match the operator's mental model of the connectivity of the power subsystem. An example of this display is shown in Figure 9.

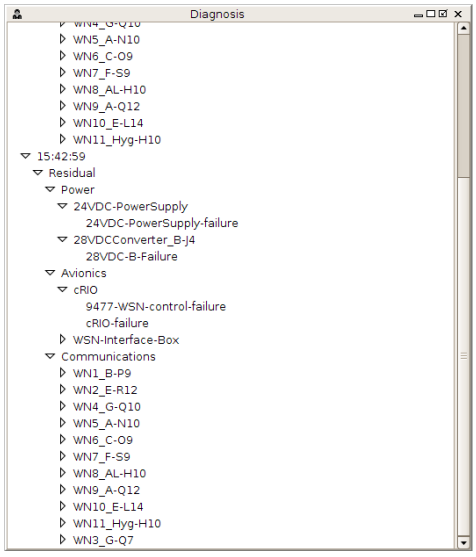

Figure 9: HDU diagnosis shown with tree display.

For procedures, we integrated with the procedure display developed by the SHIELD Discrete Controls project. It too uses a similar framework of enabling the operator to configure the display format to suit individual needs. Although malfunction procedures are recommended by ACAWS, all interaction with procedures is via this Procedure Display

(PD), shown in Figure 10. Through the PD, the operator is able to provide system observations needed to disambiguate the cause of a failure.

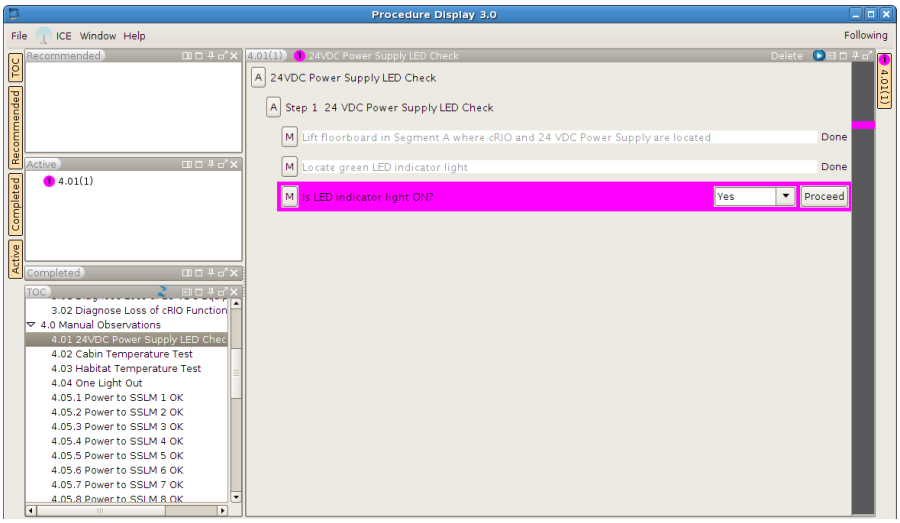

Figure 10: Procedure display.

Two diagrams were developed specifically for the HDU power string investigative domain: a block diagram shown in Figure 11 illustrating connectivity of the components involved in the demonstration scenarios and a tabular sensor display shown in Figure 12 which resembles the tabular displays MCC operators are familiar with. In a departure from most current systems summary display design philosophies, our display deliberately blended components from different HDU systems (Power, Communication, etc.) in order to provide operators with a single display format that communicated functional connections between components, and a single, integrated source of information about the functioning of the diagnostic reasoner and the full constellation of failure impacts.

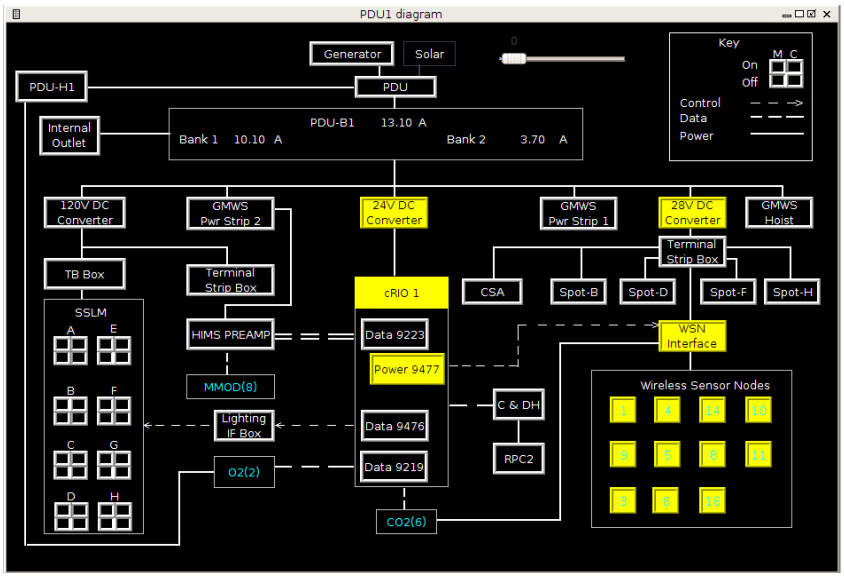

Figure 11: HDU block diagram display.

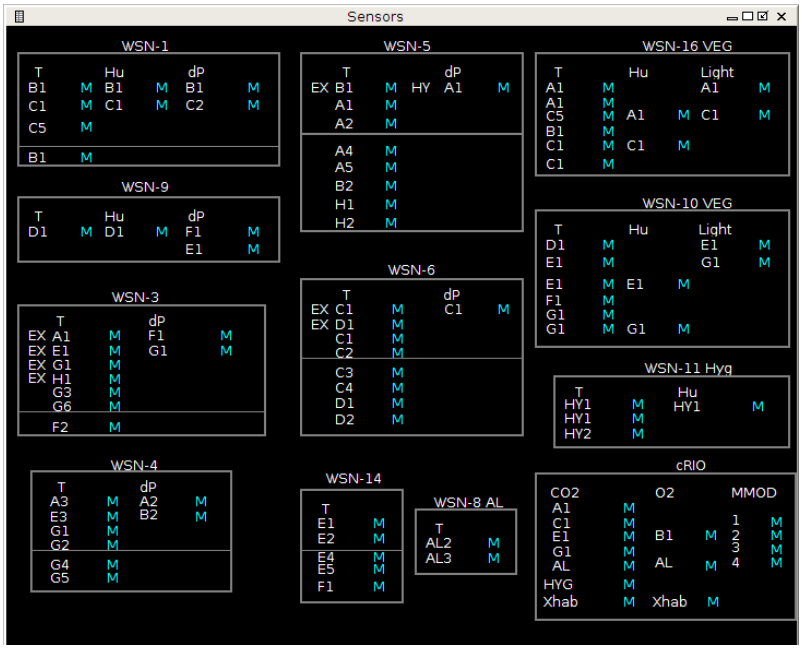

Figure 12: HDU sensor display.

User Interface Design Considerations

The ACAWS GUI is developed in Java. Display screens are programmed in Java Swing, with some GWT components, and an open-source software package – *MyDoggy* – that provides window allocation and docking capabilities. *Docking* means that a window can be dragged/dropped on a "docked" location and stay there relative to other windows, where it will move, resize, etc. along with those windows. When it is undocked, the window can once again be manipulated individually. Some customized work has been done to add features such as saving the configuration of the working environment, docking multiple windows around a main

window, and partitioning different areas of windows to group them together. The ACAWS GUI framework simplifies dynamic management of multiple working windows at run time. The ACAWS GUI has multiple types of window panes to display different types of information with different kinds of user interaction. All types of windows panes are derived from the *dockable* framework so that all windows are dockable and manageable in the same fashion.

Design of the ACAWS GUI is based on the *Model View Controller* (MVC) software architecture. The controllers manage the data aspect of the components, and are responsible for data retrieval, processing and mapping, and also accept the user's inputs. The views are responsible for updating display panes and rendering the components. The views also respond to the controller's requests. The models hold data, control, view and auxiliary information together, to notify the view to update when the state or data changes. The purpose of using such an architecture is to separate the view from the model so they can independently change or be modified, in a one-to-many relationship. For example, when the model logic changes, the view part of the code doesn't necessarily need to change. One model can have multiple views, which can help the user understand different aspects of the model. There are multiple types of window panes in the framework to display and take user action on different data types. Each of the window panes utilizes the MVC architecture. For example, the telemetry data provides sensor values that can be displayed on the sensor data pane. The same data can also be displayed on the block diagram but using a different view, in this case, just whether the data is available or missing, not the data's value. One model with two views can be realized easily with this MVC architecture.

A multithreading model is used in the ACAWS GUI implementation. The ACAWS GUI connects to the communications middleware (the *ICE* messaging server) to get data from ACAWS; this data feeds in real time. There are five data sets involved: HDU sensor readings, TEAMS diagnosis results, recommended procedures, repair procedures, and IMS cluster distances scores. ICE services the data feed using a publish/subscribe scheme. Each data feed is a *topic*, and the ACAWS GUI subscribes to each of the data streams by topic name. When data is available, the ICE server will call back to the client to send the data stream. If all the data sets are available at the same time, all will be streamed to the ACAWS GUI concurrently. The multithreading model can prevent view updating "jitters" when all data streams flood in at same time and the controller requests all views to update at the same time. The jitter situation is especially bad when multiple window panes are on one dockable window and all panes are viewable. The use of multithreading can utilize multiple CPU time slots and run multiple requests in parallel, which increases performance and reduces problems of data lag or updating lag of the displays. The multithreading model utilizes Java Threads and synchronizes using thread states of *run*, *stop*, and *suspend*. The CPU time allocation for each thread is done at the Java virtual machine level so that the ACAWS GUI does not need to concern itself with thread scheduling.

Color coding, using the traditional caution and warning system off-nominal colors of yellow and red, was used on the annunciator panel and block diagram to communicate TEAMS-derived information to the operators. Filled yellow components (Figure 8 and Figure 11) corresponded to system elements that were currently in the TEAMS suspect list, meaning a failure of the colored component was consistent with (could have produced) the current pattern of test passes,

fails, and unknowns. A filled red component corresponded to an element deemed "BAD", i.e., certainly failed. Once an element was deemed "BAD", downstream components that are currently deemed nonoperative due to the upstream failure were colored outline red, indicating the functional impacts of the upstream failure.

On the sensors display, sensor values corresponding to missing data were replaced with cyan "M''s.

Demonstration Evaluation

Three evaluators participated in the ACAWS evaluation. Of the three, two were "field evaluations", each featuring one operator/evaluator (hereafter, O/E), and each taking place on one each of two successive days at the D-RATS site. Each of the two O/Es assumed the role of an on-site operator with full responsibility for diagnosing and working procedures for each of two malfunctions that were hand-injected into the HDU electrical power system by one of two confederates, both of whom were co-located with the O/E in the HDU. As the evaluations were not conducted under conditions where the O/E had complete capability to work all the manual procedures, the role of the other confederate was to provide the O/E the correct option to select from the pull-down menu of step-appropriate possibilities (e.g., if the step asked the operator whether a light came on in response to an earlier "Turn on light" command, the menu options were "Yes" or "No").

The third evaluator was located in the D-RATS MCC at NASA JSC. The original plan was for this evaluator to be a passive observer of the ACAWS displays, similar to the ACAWS members in the onsite tent, except that telemetry transmission (and resulting display updates) would be delayed by 50 sec. This plan was overtaken by events, however, when the ACAWS telemetry failed to transmit to MCC. Accordingly, the MCC participant performed his evaluation "after the fact" using playback files, and was instructed to do his evaluation while applying the mindset that he was in fact "in the loop" on the day of the actual evaluations, but watching events unfold with a 50 sec delay.

Evaluation Methodology

Prior to participating in the actual evaluation, all three evaluators received approximately one hour of ACAWS training. The goals of the training were threefold. First, at the most general level, the evaluators were "coached" through a 28V DC power failure and the attendant ACAWS displays using PowerPoint slides. In this way, the evaluators were familiarized with the elements of the ACAWS display elements, the design philosophy behind the displays, the color-coding conventions used to communicate TEAMS diagnostic information, and the procedure recommendations from the TEAMS diagnostic reasoner to the operator. The second training goal was to familiarize the evaluators with the standard set of windows icons that they could utilize to navigate between the ACAWS displays, resize the displays, and locate them on the screen according to user's individual preferences. Finally, the third goal was to familiarize the evaluators with the set of design features (e.g., the focus bar, the window containing TEAMS-recommended procedures, etc.) and the user

interfaces (e.g., pull-down menus, "Proceed" button, etc.) of the Procedure Display (PD) that enabled the evaluator to navigate through and complete a procedure.

On each of the two test days, one of the two O/Es and the two ACAWS confederates took up residence in the HDU at the completion of that day's scheduled integrated test activities. The test protocol, which took approximately one hour to complete, consisted of two EPS failure scenarios that were injected into the HDU electrical power system manually by one of the confederates. The first failure produced an unambiguous signature that TEAMS was able to diagnose as a wireless system node (WSN) failure. The second failure, a 24 VDC fault for O/E 1 (first testing day) and a cRIO card fault for O/E 2, produced an identical set of failure suspects that could only be disambiguated to the "parent" failure by providing TEAMS with the result of an additional manual test, part of the TEAMS-recommended procedure. Each O/E was initially expected to perform all the manual steps involved in the procedure. For example, the disambiguating test for the 28V DC failure (the scenario used in practice) involved turning on a spotlight that was powered through the 24VDC line, and then visually checking the spotlight through a camera to determine whether the "turn on" command had succeeded (and the spotlight was in fact on). In order to make that assessment, the appropriate camera had to be located and slewed to an orientation that brought the spotlight into the camera's field of view. In the original plan, the O/E would have actually performed these actions using the dedicated camera display brought up on an iPad, one of a suite of non-ACAWS displays designed for D-RATS operators. However, the HDU did not turn out to be configured to support these actions directly, so one of the confederates coached the O/E to advance through the manual procedures, and then whether he should select "Yes" or "No" on the pull-down menu in response to the step that asked the crucial question, e.g., "Is the cRIO LED indicator on" for the cRIO/24VDC failures, in order for TEAMS to disambiguate these failures.

Even with this assistance from the confederate, the O/E was responsible for diagnosing the source of the failure using the information provided on the ACAWS displays. Following the completion of each of the steps on the PD, the O/E completed a questionnaire consisting of a workload rating (Bedford scale), and ratings of ACAWS display characteristics. Following the completion of the procedures appropriate for the second failure, these scenario-specific questions were re-administered. Finally, a more general set of ratings and questionnaires pertaining to the ACAWS displays were completed. In addition to these quantifiable metrics, ample opportunities were provided for the O/E's to give us more unstructured feedback (opinions) as to the content of the ACAWS displays, whether the level of integration between the ACAWS displays and the PD was sufficient, what kinds of additional capabilities the evaluators would like to see from ACAWS in future iterations, and so on.

All evaluation results are presented and discussed in *User Opinions* section below.

8. Results

Diagnosis+Procedure

The diagnostic results from the ACAWS field trials were analyzed for correctness and effectiveness and a full report of the analysis is presented in a separate document. This report summarizes the diagnostic results as well as the key findings of the analysis.

Diagnostic Results and Correctness Assessment

The HDU model with HDU data produces correct diagnoses, either isolating to the correct inserted failure, or to an ambiguity group that includes the inserted failure. The reasons for an inability to resolve to a single component are understood and explained. The diagnostic results were accurate representations of the state of the HDU systems once the failures were fully elaborated. In all cases the correct diagnosis of the seeded failure was represented within the body of information. The Standard Diagnosis was very consistent and presented a correct diagnostic representation. The Minimal Diagnosis, which was used as the basis for information displays, was somewhat susceptible to perturbations from anomalies in the data.

A summary of each scenario that was exercised in the field is provided in Table 3.

Table 3: Diagnosis Summary

Scen	***Failure***	***Standard Diagnosis***			***Minimal Diagnosis***		
		Bad	***Suspect***	***Correctness***	***Min***	***Resid***	***Correctness***
1	WSN-5	0	9	Correct ambiguity group; seeded failure is included in SUSPECT	1	0	Correct isolation to seeded failure.
2 A	24 VDC Power Supply	2	89	Correct component in BAD, 1 sensor also in BAD Correct SUSPECT	2	9	Seeded failure in MIN, 1 sensor also in MIN RESID was not cleared
2B	24 VDC Power Supply	1	91	Correct BAD Correct SUSPECT	1	0	Seeded failure isolated as root cause
3	9477 Control Card	0	87	Correct SUSPECT but incomplete resolution; additional manual tests needed.	0	87	Identical to Std
4	cRIO	10	79	Seeded failure in BAD, plus 9 sensors	3	0	Seeded failure in MIN, plus 2 sensors

				Correct SUSPECT set			

The diagnoses were consistent with the data and the model, and where the seeded failure was not isolated as the single failure, the causes for ambiguous information are explainable. In Scenario 2 a software modification after the evaluated test run provided a more correct diagnosis. In Scenario 3, an additional manual test that was not performed would have resolved the ambiguity. Scenario 4 was nearly correct but was affected by incomplete initialization after the previous scenario, but the displayed information was correct.

Standard Diagnosis left large SUSPECT sets in addition to usually including the correct failure in the BAD set. An examination of the SUSPECT set shows that the components are justifiably included based on an assessment of the HDU system and the diagnostic algorithm.

Summary of Findings

The primary findings of the field exercise analysis are briefly summarized, and additional detail and explanation is provided in the following sections.

- Finding 1 – Synchronization and timing of HDU data causes variability in the sequence and presentation of information during failure onset, but produces consistent final results.
- Finding 2 – Frequent data anomalies including false INVALID indicators, false FAIL and false PASS results during failure onset, and false heartbeat FAIL tests result in intermittent false diagnostic results but in most cases the correct diagnosis is restored when the false test results are cleared.
- Finding 3 – Inconsistent and invalid cRIO sensor value tests during failure onset affected the diagnosis in scenarios 2 and 4 when cRIO function was lost.
- Finding 4 – The use of Minimal Diagnosis as primary display is useful when diagnosis resolves to a single root cause and the RESIDUAL set is cleared, but the RESIDUAL set as the basis for diagnostic display is less than satisfactory. The SUSPECT set from Standard Diagnosis is more stable. Further analysis is recommended for selection of diagnostic algorithm.

Synchronization and Timing

The HDU was developed as a concept demonstration system with subsystems that are generally representative of spacecraft. The HDU Avionics team made an architectural decision to use a loosely coupled data system without rigorous synchronization. Data cycles would be based on individual system clocks, without particular requirements for time synchronization, jitter or drift. Software applications would be responsible for controlling their execution cycles, and if the processors were busy when the application was ready to execute, delays were tolerated. The avionics team worked to keep processing times reasonably efficient and keep network bandwidth moderate. Rigorous control would have required a degree of design, development and analysis that the group could not attain with the limited budget and schedule available. A habitation system like the HDU does not

necessarily require a rigorously deterministic real-time controls system, since most response decisions can be made slowly, it was believed, and inconsistencies at the transitions can be tolerated. The architectural decisions were made after consideration of benefits and risks, and are not being second-guessed here. This type of system, however, provides additional challenges for an automated decision system, and these characteristics must be considered early in the architecture of the health management system.

Humans are generally pretty good at recognizing when data comes in slightly out of sequence, and can mentally process the data to understand its meaning. Computer systems are dramatically inferior in making these types of judgments, and in the ACAWS project, integration and performance tuning were significantly affected by timing and sequencing inconsistencies in the HDU data. In model design, it was expected that when the failure occurred, all the data would arrive at least in sequence, if not precisely at the same time. In reality, failures on the HDU system develop over an extended period. From the first indication of failure to arrival of the last element of the failure signature could take up to 80 seconds for the complex failures that involved many sensors at a mix of processing rates. These contradictions during failure onset must be expected and accommodated. ACAWS modelers did not have a good understanding of these types of issues at the start and struggled to accommodate them.

One suggestion has been to develop a rigorous framing system that assures a consistent failure signature at all times. While this would certainly help solve this problem, it has a cost in complexity of development. Developing a framing application independently of the HDU may be feasible, but achieving determinism in the failure onset sequencing could be difficult, as the full range of the HDU performance in the absence of rigorous timing requirements is only roughly known and empirically observable. Building such a system based on observed behavior may not be satisfactory; even if the frequency of occurrence of contradictory failure signatures is reduced but not eliminated, the automated diagnosis and decision system must still be tolerant of deviations, or the operators must be tolerant of errors in the system.

Data Anomalies

Frequent data anomalies are present in the HDU data from various causes. The diagnoses depended on use of validity indicators in the data, which were based on a “staleness” algorithm, not a true validity check. HDU sensors are processed and reported at periodic intervals based on clocks in the data collection equipment, and processed at periodic rates by HDU software applications, but various clocks are not synchronized and there is no attempt to accurately frame the data. When a software RIU application reads sensor data, it checks to determine if new data has been written since it was last processed, and if the data is not fresh, it marks the data as INVALID. Variations in clock drift and process scheduling can cause data to be missed which results in false INVALID settings. These false failure indications occur somewhat regularly, with some sensors more susceptible than others. The PDU-B1 total current sensor, for example, has a false failure rate of approximately 17% in the September 4 data log, while the Bank 2 sensor has a false fail rate of about 3% and the Bank 1 sensor had no false fails. These randomly occurring false fails induce false diagnostic results that appear in the displays for a period, and then clear when the false fails clear. Because of the slow data rates that range from 10 seconds to 60 seconds, the

false fail tests result in diagnoses that can appear in the displays for up to a minute. The data latching that was done to work around some display and diagnostic concerns compounded the effects of false fail tests.

Inconsistent Transition Performance

One case was observed where variations in the test results during failure onset had an impact on the final diagnosis. The CO_2 and O_2 sensors attached to the cRIO were tested in two ways. The validity tests from the RIUs that processed the cRIO data were set to VALID if the cRIO transmitted any data, and were INVALID if no new data had been obtained from the cRIO. A different test was performed by the HDU Software Controller application on the value of the sensors. This test was intended to detect a non-functioning or malfunctioning sensor by testing for large negative values that were far outside of normal operating range of the sensor. However, when the cRIO failed, the sensor value tests were expected to produce an UNKNOWN result because the Controller checks the Validity Indicator before performing the test. However, during failure transition the Validity Indicator was not set to INVALID for one or two cycles of the SW Controller, resulting in false test results published by the SW Controller. In 7 of 8 of these tests the results were always FAIL results, but the Airlock O_2 sensor test varied between PASS and FAIL from different test runs, and flipped between PASS and FAIL in at least one case during the transition. These test results were passed to the diagnostic reasoner and caused a slightly different ambiguity group as a result. When all the tests were FAIL, the cRIO 9219 Card was implicated as a BAD component; when one test was PASS, the 9219 Card was not implicated and instead the sensors that reported FAIL sensor value tests were diagnosed as BAD. The variability in the B1 O_2 sensor had an effect on the displayed state of the 9219 Card that was confusing to operators and was not easily explainable.

Results varied between the three test runs in which cRIO functionality was lost. Two runs of Scenario 2 were conducted on 9/4 in which 24 VDC Power Supply failure resulted in loss of cRIO function, and one run of Scenario 4 on 9/5 where a cRIO failure was induced. The diagnosis of the 9219 Card varied between these three based at least in part on the random variability of the test results on the cRIO sensor values. A means of suppressing the false test results after cRIO function is lost would have eliminated all of these effects and made for a more accurate diagnosis especially in the transition states.

Diagnosis Method

The QSI TEAMS-RT tool includes two diagnostic algorithms, Standard Diagnosis and Minimal Diagnosis. The ACAWS team has considered both, and decided to use the Minimal Diagnosis as the basis for displays. The ACAWS project has provided a substantial body of data that can be used to make a much more complete and analytical assessment of the preferred diagnosis method. Further analysis using the data from HDU failure scenarios is strongly recommended for future health management projects.

Standard Diagnosis uses a Multiple Fault Assumption in which diagnosis is done without making any assumptions about probabilities. In the HDU scenarios, failures of individual components are considered as likely as a higher-level component failure that caused the test failures. For example, if 10 sensors on a WSN report test failures at the same time, the

Standard Diagnosis algorithm assumes that each of the 10 sensors may have failed and that the WSN has also failed.

Minimal Diagnosis uses a generalized Single Fault Assumption that tries to find common cause for the test failures, and if a single failure can explain all of the test results, then only the common explanation is displayed. In the WSN example above, Minimal Diagnosis concluded that the WSN has failed and the sensors have not failed.

QSI describes the Minimal Diagnosis as:

> "The Minimal Fault Assumption is a generalization of single fault assumption – it is the minimum number of failures that explain the signature (i.e., Pass/fail results). No enumeration/evaluation of failure combinations are done as that will be NP-hard.
> The biggest difference between single fault assumption and minimal fault diagnosis is that under single fault assumption, when a test fails, all faults that the test does not detect has to be declared GOOD! That is a hugely optimistic step – most people do not realize it – but that's what single fault assumption means."

The cascading failures typical of power and instrumentation systems that the ACAWS team selected for diagnostic scenarios generate lots of downstream effects, and it seemed that Standard Diagnosis would generate large ambiguity groups that experience or intuition would conclude could not be the explanation. In some scenarios there are 80 sensors, several Wireless Sensor Nodes, power supplies, junction boxes, and avionics components that are affected by power failures. It would not be reasonable to believe that a possible explanation for the failure signature is that all of those components failed, independently, at the same time. It was expected that Minimal Diagnosis would do a better job of identifying the best explanation of the failures.

The scenarios that were selected are characterized by a loss of visibility into system state due to the loss of power. The diagnosis uses the loss of visibility as the primary symptoms of the failure, but using loss of information as the diagnostic cues presents some challenges. The HDU does not have redundant power or instrumentation systems that could provide confirming indications of failure. There is power system instrumentation at the PDU level, but no instrumentation of power beyond the PDU in the DC power systems. Because of the sparseness of direct sensory indications of failure, the Minimal Diagnosis appeared to be a favorable diagnostic algorithm.

This decision warrants further consideration. The Minimal Diagnosis algorithm is not as mature and robust as the Standard Diagnosis, it appears; QSI has cautioned against over-reliance on it. It is more of a spot-check algorithm without (or less, maybe) use of prior diagnostic history and memory of prior tests, so is more susceptible to data anomalies such as occasional false failures which the HDU data is prone to generate. On detailed examination of the diagnostic results, it does not appear to be as effective as expected in thinning the diagnostic results compared to the Standard Diagnostic algorithm.

On the plus side, the Minimal Diagnosis was effective at identifying the actual failure and clearing the SUSPECT set from the Standard Diagnosis. When the disambiguating manual procedures were performed, a single component could be displayed from Minimal Diagnosis, since the RESIDUAL diagnosis was either cleared or substantially reduced. In

Standard Diagnosis even when the seeded failure was identified as a BAD component, the SUSPECT set remained large.

However, the Minimal Diagnosis is affected by conditions not directly related to the diagnosis. For example, the effect of occasional data dropouts had a somewhat surprising effect on the Minimal Diagnosis. During the HDU tests, an unexplained loss of data for about a minute occurred on two occasions. The dropout occurred when the seeded failure was fully elaborated and stable, and a stable diagnosis would be expected. When the dropout occurred, the software heartbeat tests failed, and the diagnosis correctly identified the software applications as having failed, placing them in the BAD and MINIMUM diagnosis sets. However, when these were added to the MINIMUM set, all of the components in the RESIDUAL set were cleared. When the data resumed, the heartbeats reported PASS results and the RESIDUAL set returned to the original state before the data dropout. The Standard Diagnosis was not affected by the data dropout, other than to add the software modules to the BAD set, and clear them when data resumed. Using the RESIDUAL data as part of the display results in information changes that are not explained by the condition of the actual failure.

This is by no means a complete analysis and comparison of the two algorithms. It does indicate that such an analysis should be conducted in future projects. It may be that both are useful algorithms and can contribute to a mature diagnostic display. The Standard Algorithm is perhaps a more pessimistic one, and the Minimal Diagnosis is more optimistic. An ability to mode switch between them could be useful, although it would require a more nuanced understanding of the reasoners to determine when one or the other is more appropriate.

Additional collaboration with QSI is recommended both to more fully understand the two algorithms, and to possibly refine and mature the Minimal Diagnosis algorithm.

Procedures

Currently procedures for the HDU are written *offline*, i.e. they are authored and validated in advance of their usage in mission operations. Support for dynamic in-situ procedure authoring and modification exists but needs further refinement and testing before actual use. Procedure validation of a static nature is currently performed in the PRIDE authoring tool, which checks procedure content against the HDU system model (represented in XTCE) as the procedure is entered; violations are immediately reported. Dynamic verification of procedures is also achievable in PRIDE by running them against a state machine representation of the HDU; this is not currently performed due to insufficient development of the needed state machine representation.

All procedures written for the HDU field test proved to be effective in accomplishing their goals. We note that the repair, malfunction, and manual observation procedures were almost completely *manual* procedures, i.e. containing instructions that had to be carried out by a human. Because these manual steps were well known in advance, the procedures had high likelihood of succeeding. Some automation, in the form of automated commanding and telemetry checks, was added to several repair procedures for the final live HDU test, and the automated execution was also successful.

User Opinions

Field Operators. When considering the feedback provided by the two ACAWS O/Es, it is important to note their divergent backgrounds: O/E 1 is an advanced systems development engineer and D-RATS operations manager, while O/E 2 was a former mission control console operator. In addition, the two O/Es completed their evaluations under different circumstances. On day 1, TEAMS encountered difficulties disambiguating the suspect list, so even after the second failure was disambiguated to the parent problem (24 VDC failure), a list of residual suspects remained. On day 2, software improvements allowed the TEAMS System Display to eliminate the residual suspects, showing (correctly) only the parent failure (cRIO card) in filled red. As we will see shortly, O/E 2 returned consistently higher ("better") values with respect to subjective evaluations of the ACAWS displays and specific elements thereof than O/E 1. With the background of the O/E's fully confounded with these test protocol differences, it is not clear whether to attribute this systematic result to occupational background differences, the different conditions at test, or even just to random (and expected) differences in how individuals interpret and fill out Likert and Workload scales (for instance, individuals often bring different thresholds and preferences for selecting extreme values close to the anchor points on the scales to their evaluations). Therefore, our analysis of user feedback will focus for the most part on general trends rather than individual differences.

Workload. On both malfunctions, workload was judged as a "2" or "3" on the Bedford scale, with the exception that O/E 2 selected numbers closer to the middle of the scale for the isolation and recovery stage. Overall, therefore, the O/Es self-rated their workload as fairly light, with plenty of spare capacity to deal with additional tasks if they had to, at least during the diagnostic phase. There was some indication that more attention was required, and there was less capability to deal with additional tasks, while working task isolation and recovery procedures. The MCC console evaluator (hereafter, "C/E") did not rate his workload.

Information Display. Following each scenario, all evaluators were asked to choose a value between 1 and 10 that corresponded to their assessment of how timely the display of information was on the ACAWS displays, how well the information depicted system state, how well the information supported a quick and rapid assessment of the malfunction, how well the displayed information enhanced their decision-making capabilities, and how well the display of information helped them complete the necessary procedures. The results for most questions were generally in the high 7-10 range, indicating quite positive assessments. A notable exception was that O/E 1 gave a "5" (Scenario 1) and "6" (Scenario 2) to the question about how well the displays enhanced decision-making capabilities. This relatively low assessment probably reflects the problems with ACAWS information presentation for O/E 1 compared to O/E 2. At a minimum, the data point to the importance of unambiguous color-coding on the ACAWS displays for operator usage. All but one rating (by the C/E) were 7's.

Ease of Processing/Ease of Operation. As a corollary to the Information Display-related questions, which tied operator's performance to the information on the displays

themselves, evaluators were also asked a series of more general situation awareness-related and display-usage questions, such as, on a 10-point Likert scale where zero was very easy and 10 was very hard, how easy was it to understand the source of the problem, and to engage in activities such as locating the appropriate information on the ACAWS displays and working the malfunction. Again, the responses by the two O/Es clustered in the low end of the scale, with an overall average across eight individual ratings (four for each scenario) of 1.75. Consistent with the results from the other assessments, the trend was for Scenario One to be rated easier to deal with than Scenario 2, and O/E 2 rated the scenarios as easier than O/E 1. The average of the C/E's ratings was 2.25, also well toward the "Very Easy" side of the scale.

Display Format Features. Following the completion of the two scenarios, evaluators were asked to rate, on a 10-point Likert scale with 0 being "Not at all Favorable" and 10 being "Very Favorable", three general questions concerning the ACAWS block diagram display. The questions were all related to the overall issue of how well the "show the intra and inter-systems connections" philosophy that was the primary driver behind the design of the block diagram display "went over" with the O/Es. In general, the O/Es gave quite favorable feedback with all responses on the "Very Favorable" side of the scale, and an average rating over the six total responses (three questions, two O/Es) of 7.8. Once again, the ratings were slightly but consistently lower for O/E 1 than for O/E 2. Ratings by the C/E averaged just under 7, also comfortably in the "favorable" range.

A second series of questions concerned the usefulness of specific features of the block diagram display on a 10-point Likert rating scale with 0 being "Not at all Useful" and 10 being "Very Useful". A key was presented in the upper right-hand corner of the display to give "at a glance" information concerning coding conventions adopted for the display, such as the fact that connections were solid lines when depicting power buses between system components, long dashed lines for data buses, and short dashed lines for commanding buses. One O/E declined to rate this feature; the other rated the usefulness as 8.75, close to the top of the scale; the C/E returned a slightly less favorable rating of 6.75. Turning to the more general question of how intuitive the ACAWS color coding conventions were in and of themselves, on a scale of 0 (Not at all intuitive) to 10 (Very intuitive), O/E 2 though the coding was a highly intuitive 8.75; O/E 1 and the C/E rated the coding as just slightly less intuitive, giving it a 7.25. In general, then, both the overall design format and several specific features of the block diagram were well received by all three evaluators.

Other displays. O/E 2 rated the usefulness of the two remaining ACAWS displays (annunciator panel, sensor) as 8.75 out of 10. Continuing the trend from other questions and other evaluations, O/E 1 was somewhat less charitable, giving the annunciator panel a (still positive) 6.25, and the sensor panel at 7.75. Both O/Es rated the windows-related ability to re-arrange and resize the displays at close to the top of the usefulness range; 8.75 in both cases. With regard to specific design features, the focus bar received a rating of 8.75 by both O/Es on the usefulness scale. On the other two features, the small magenta position bar and the pull-down menus for user inputs, O/E 1 again rated the magenta position bar as less useful than O/E 2; both thought the pull-down menus for

communicating the results of user actions to the PD were quite useful (8.75 and 8.25, respectively, for O/Es 2 and 1). For his part, the C/E rated the annunciator panel, the schematic display, and the ability to rearrange the panels to suit his individual preference all above 7.0 on the usefulness. However, in a sharp departure from the O/Es, the C/E only gave the sensor display a 3.75 rating, and called the display not very useful in his comments section. He thought that the display layout overemphasized WSN sensor data to the detriment of the other sensors depicted in the block diagram display.

Interestingly, both O/Es had additional comments on what additional features they would like to see in a PD. (The C/E did not see this display, and did not provide PD-related input.) O/E 1 wanted the steps to be numbered; O/E 2 suggested a time stamp on recommended procedures, possibly as a tie-in between the PD and the astronaut's activities timeline. An ability for the operator to clear and/or delete procedures and a list of recently completed procedures and status in table form were also suggested.

Integration of PD and Block Diagram Displays. The ACAWS developers had a particular interest in getting more information from the evaluators concerning the degree of integration that was built in between the PD and the Block Diagram. We have implemented various concepts in the past that feature more integration than in the current ACAWS concept, where integration was limited to the ACAWS displays recommending procedures, and the PD containing a window featuring the same list of recommended procedures. Here the interesting result was an interesting dissociation between the views of O/E 1 and 2. In a deviation from the normal pattern, where O/E 1 yielded lower ratings than O/E 2, O/E 1 rated the current level of integration between the two displays as higher on the "0 equals insufficient and 10 equals sufficient" scale than O/E 2. Consistent with that assessment, O/E 1 chose a value on the negative side (4.25) of the scale in response to the question "How useful do you think it would be to try and combine PD information with system status and failure impact information on the same display", whereas O/E 2 though it would be very useful (rating of 9.75). This was by far the biggest dissociation in ratings between the two O/Es, and was reinforced by O/E 1's comment that "Care should be taken when trying to integrate procedures too closely with other failure data". By contrast, O/E 2 suggested grouping procedures by subsystem.

Additional Features and Capabilities. All three evaluators were asked how desirable it would be to add the following elements and capabilities to the ACAWS system: Impact assessments that included impacts on activity schedules and operator timelines, automated, real-time procedure generation, a natural language-based interface for commanding, troubleshooting, and querying databases; and a prognostics (time-to failure) capability. While O/E 2 consistently rated the desirability of these possibilities higher than O/E 1, they both rated them as quite desirable, with the lowest rating being a 7.25 from O/E 1. Similarly, the lowest value provided by the C/E was 7.75, for the natural language interface.

IMS Assessment. O/E 1 and the C/E were not able to get exposure to the IMS displays during his evaluation period. O/E 2 rated the IMS displays as a 7.75 out of ten on a rating

of how useful he found IMS for maintaining situation awareness of the overall health and functioning of the subfloor area, 7.25 out of 10 for how intuitive he found the depiction of the aggregate deviation score, and 7.25 on a scale of 0 (Less Useful) to 10 (Much more useful) an aid to situation awareness that he found the depiction of the aggregate deviation score to be, compared to the standard depiction of individual sensor values color-coded for limit-sensing.

Overall comments for additions to ACAWS. O/E 1 suggested adding percentage/probability indicators to the suspect list and embedding the indicators in the block diagram alongside the yellow components. O/E 2 suggested that ACAWS be able to generate a list of possible failures before any procedures are run to help situational awareness for operators. The C/E recommended larger console monitors, or multiple monitors, so all ACAWS displays could be in the field of view at once. In addition, he thought that the displays need to have a clearer way of communicating TEAMS behavior and TEAMS classifications (i.e., what constitutes being part of an ambiguity group) during the time that a suspect group is being depicted (i.e., prior to declaring an element "BAD"). This led him to suggest that ACAWS should consider being more selective, such that not all failures and anomaly situations may require the information about them being filtered through an automated tool like TEAMS.

C/E evaluation of time delay effects. While viewing the ACAWS displays and watching them change as the failure was injected and TEAMS was processing the data and making its classifications, the C/E was asked to evaluate from the perspective of passively watching the situation unfold with a 50 sec time delay relative to the field. He deemed the questions N/A, however, citing the fact that he had no responsibility for working the HDU malfunctions and did not collaborate with the crew in any way. As he stated, "Time delay will only have an impact to ops if the task is a critical task or if the crew does not know how to handle a situation. Automation is needed in order to assist the crew with situations like this."

His assessment of, if he had had a role to play, what the impact of a 50 sec time delay on that role would have been, was in the "no discernable impact" range of the 10-point scale.

9. Lessons Learned

As with most endeavors, there is always room for improvement. In this section, we discuss some of what we learned that could benefit future projects.

Development process:

- Follow the agile development process as intended. Mini-deliverables along the way help determine whether adequate progress is being made toward the proposed deliverable. It helps ensure that the customers' top priority requirements are met early in the process, allowing guidance toward what was really wanted, not just what they thought they wanted. It also allows the development team get to the

hurdles more quickly and have the time available to work out a method over the hurdle or transition toward a different solution before too much is "laid in stone."

- Integration planning. Didn't plan sufficiently for how to test, other than hook up to HDU and run. The team struggled with lack of test tools throughout integration, cobbling together a playback feature late in the game. HDU Tester was OK for static interface testing but incapable of end-to-end scenarios that was desperately needed. Several issues encountered on the real hardware were impossible to re-create using the playback tools at hand. Development of a higher fidelity and complete simulation model to test against would have greatly facilitated code design and reduced stress.
- Develop modules such that they can be thoroughly unit-tested, i.e. not highly dependent on completion of other modules. Have ICDs agreed upon early.
- Select a system architect and development lead at the *beginning* of the project, someone who will coordinate and facilitate testing and integration through the final test. Ensure that person has the required skills to perform the task.
- Acquire needed hardware and third party software, and have it set up, well in advance of the field test. Back-ordered hardware caused unnecessary work, especially when the effort of complicated software installs and licensing had to be duplicated on multiple machines and architectures. Do not assume that just because a software installation process was worked out on a similar computer/OS that it will work flawlessly on the target computers/OS. Do not put off that install and verification until the end of the development process; people will be too busy finishing up their tasks to support that task.
- Customize the user environment(s) on the field test computers so that they are easy to use by all team members involved. Ensure that all team members are part of the same permissions group and that the "umask" is set appropriately to allow everyone access to code/data files.
- Streamline the running of the project's software. If multiple programs need to be started or a particular start sequence needs to be followed, that process should be simplified as much as possible, perhaps even fully scripted, and definitely fully documented so that everyone on the team can perform it.
- Creation of a group account for the team would be useful in advance of the demonstration, and should be tested on all demonstration machines prior to the field test. This may not be allowed by security restrictions, however, so work-arounds should be considered in advance of field testing.
- Develop software, especially GUIs, on the target operating system (OS). Porting from a different OS adds tasks and even for software promoted as OS-independent, variations between OSs will occur. This is especially true for the GUI. The look and feel on a Windows computer can be very different than the look and feel on a Linux computer. Elements will need to be resized to look right on the screen – effort that could be used to develop additional functionality rather than redo work.

Demonstration domain integration:

- If the demo domain hardware or software is not already fully developed, assume and plan for development of demo domain hardware and software needed for integration to slip. Develop as much technology that does not depend on the demo domain, i.e., the general functionality, as early as possible. Make assumptions about how things will eventually work in the demo domain hardware and avionics but frame those assumptions in the code so they are easily replaced.
- If the demo domain hardware or software is still under development, get agreement on interfaces as early as possible. Make project requirements known to the demo domain hardware/software team as early as possible and work together toward reasonable solutions that address both team's constraints. Case in point from this year's effort was the misfit between our need for validity flags that provide information on whether a telemetry value should be trusted versus the avionics' team implementation of a "freshness" flag that provides information on how recently the sensor was polled for a given telemetry item.
- Chasing a moving target. The HDU was continually changing through integration. The first time ACAWS was ever run with a complete correct software load on the HDU was in the field demo. The lesson is not necessarily to avoid working with articles under development, but when doing so, planning needs to account for the churn. Test tools, expectation setting, using ACAWS as part of the design process all would have helped.
- Schedule integration testing time with the demo domain hardware/software before the field experiment. There were many concurrent experiments at D-RATS that required a nominal HDU configuration and the support staff's attention. Having to do integration testing under those circumstances was very difficult. Our scheduled integration testing slots were usurped by repairs that needed to be accomplished for the higher-priority "integrated test" happening the following day. Our slot was of limited duration, allowing for only a couple of trials per slot. Because HDU time was in short supply, we were limited to only two one-hour test slots. Given that the HDU avionics was not in the correct configuration during the first one-hour test, we were left with trying to test everything out in one hour with no retest time before the evaluation runs.

Diagnosis:

- Develop with the assumption that some hardware will be failed in the field. When designing demo scenarios, build enough of the surrounding foundation to ensure the scenarios are not too fragile and unable to be accomplished when associated components that are not part of the scenario fail as they inevitably will at the most inopportune time.
- TEAMS' Minimal Fault Assumption ("minimal diagnosis") vs. Multiple Fault Assumption ("standard diagnosis") algorithms both have some strengths and weaknesses; we hypothesized that for the selected failures the Minimal Diagnosis would provide a better basis for display. The hypothesis was, at best, partially borne out.
- TEAMS "minimal diagnosis" functionality works differently than intuition may suggest. In particular, when a few things could explain a test result signature, each

of these things is put in the "residual" list rather than listing these all as possibilities in the "minimal" list. For example, if component-1 failure or component-2 failure results in the same test result signature, the minimal list is empty and the residual list contains both component-1 and component-2.

- TEAMS "minimal diagnosis" does not keep a history of test results. Standard diagnosis does keep a history. With history, TEAMS maintains its own view of the state of the system and its diagnosis is not quickly swayed by transient incoming test results. In order to provide TEAMS with an accurate view of the current state of the system, that capability must be reproduced in the diagnosis executive. Full vectors of test results must be passed in at each iteration. Conversely, when using "standard diagnosis," only newly received test results need to be passed in since it maintains the previous values for any test results not passed in.
- Timing and Synchronization. Diagnostics with non-real-time unframed data at multiple slow rates, and "ratty" data (frequent intermittent false test data) will result in a diagnosis that takes longer to reach a consistent state. While the challenges involved in programming to such a system add greatly to the body of "lessons learned", the positive impact on observers and especially stakeholders is somewhat lessened.
- Displays based on a worst-case initial assumption that is reduced over time, rather than building up to a diagnostic conclusion; e.g. if one sensor fails, the initial diagnosis is that it could be caused by anything from just the sensor all the way to a total power failure. This pessimistic view is sometimes at odds with the expectations of controllers used to more optimistic assumptions.
- Testing Paradigm. The team made an early decision to use HDU controller tests as the primary basis for tests to TEAMS. The reality was that the tests that the controllers were able to perform were not sufficient for diagnosis. Validity and heartbeats were primary data for determining if sensors and other equipment were operating. When controller tests were used, they were frequently out of synch during transitions and resulted in misleading results. A test layer outside of the HDU core software could have more effectively managed validity and other testing issues; the "valid … static" not above is part of this topic.
- Targeting diagnostic domains early. There are various flavors of diagnosis that can vary significantly in how to approach them. The power/instrumentation systems that were targeted relied heavily on sensor validity and software heartbeats, with only a few tests on telemetry values used. Software development had assumed a primary reliance on using tests on valid telemetry so some key software components were not even identified until late in the project. Other domains would presumably have quite different needs; ECLSS, with emphasis on fluids and chemistry; propulsion, characterized by high energy and rapid failure propagation. Understanding a targeted range of domains before completing the software architecture would have helped.
-

Anomaly Detection:

- Given that we received a pseudo "freshness" / pseudo "validity" flag for each telemetry parameter, it would have been useful to include rate of change as a computed parameter in the monitored vector.
- Developing the needed knowledge bases was quite easy. If data is available, the KB development tools make "cranking" out a KB and tuning it easy enough for the non-expert.

Procedures:

- Author, verify, and validate procedures well in advance of the field test.
- More study of the diagnostic model and its impact on the "Next Best Test" functionality. The recommendations provided by TEAMATE are based on diagnostic trees, but little to no time was spent evaluating these generated trees and their applicability for the demonstration scenarios. Completion of this work would have required either more time between model completion and demonstration or additional personnel.

User Interface:

- There was broad agreement on the part of the evaluators that a design philosophy for an ACAWS display that provided a graphical representation of inter-, as well as intra-systems connections between components that are impacted by a systems failure was a good design feature. This suggests that a scaled-up dynamic display generation capability that would produce a fault-management display for any arbitrary failure in any arbitrary system should be pursued.
- The unanimous, highly favorable ratings of the desirability of all four suggested additions – incorporating crew activity schedules and timeline information into the ACAWS impact assessments; an automated, real-time procedure generation capability; natural-language based user interfaces; and a prognostics capability – should all be pursued.
- There were consistent expressions of concern over the delays between the initial depiction of the suspect elements on the block diagram, and the final determination of which elements were failed and which were just impacts. The ACAWS displays clearly need to display more information to the user about what state the automation is in; how "certain" the automation is that the current depiction is the "final answer", whether the diagnosis process is continuing and the user should "stand by" while more test data comes in and is processed, etc. Another approach to this problem is to go back to the telemetry generation process and impose top-down requirements for telemetry update rates that are standardized across sensors and command and data-handling systems hardware. In addition, smarter diagnosis executive algorithms need to be written to implement a user-centered filtering scheme, where transient sensor dropouts and the like, and the immediate TEAMS response, is withheld from the displays (or, at least, the TEAMS online derivation of BAD and SUSPECT elements is inhibited).
- An important lesson learned stems from the disagreement between the O/Es concerning the value of pursuing design options for ACAWS displays that more closely integrated the procedure display and block diagrams. The difference in

opinion suggests that one suite of ACAWS displays may not fit all, and those with different roles to play in future missions (crew, flight controllers, multi-purpose support room [MPSR] personnel, etc) may benefit from customized design suites tailored to fit their particular information needs.

- The feedback from the C/E about time delay indicates quite clearly that to be meaningful, a study of the effects and impact of time delay needs to incorporate an operational concept and a failure scenario that includes a meaningful role for the ground in cooperation with the crew.

Appendix A: ACAWS Team

Management Team	Role	Affiliation
David Alfano	ASA Project Manager	NASA ARC
Mark Schwabacher	IVHM Principal Investigator	NASA ARC
Alan Crocker	ACAWS Task Customer	NASA JSC

ACAWS Development Team	ACAWS Role	Affiliation
Gordon Aaseng	Diagnostic models, scenarios, procedures	NASA ARC @ JSC
Vijay Baskaran	Diagnostic executive, anomaly detection	SGT @ NASA ARC
Gary Dittemore	MCC Domain expert, MOD customer	NASA JSC
David Iverson	IMS models/consulting; HDU failure injection	NASA ARC
Jeremy Johnson	Diagnostic models	SGT @ NASA ARC
Charles Lee	User interface	SGT @ NASA ARC
Sotirios Liolios	MCC Domain expert, HDU Integrated Ops Manager, MOD customer	NASA JSC
Robert McCann	User interface, scenarios, procedures	NASA ARC
John Ossenfort	Diagnostic executive	SGT @ NASA ARC
Peter Robinson	Diagnostic models, data system, procedures	NASA ARC
Lilly Spirkovska	Task lead, user interface, anomaly detection	NASA ARC

HDU / DC Team	Role	Affiliation
Daniel Carrejo	HDU, failure injection trainer	NASA JSC
Mike Dalal	DC	SGT @ NASA ARC
Chuck Fry	DC	SGT @ NASA ARC
Larry Garner	HDU, DC, integration	Tietronix @ NASA JSC
Matthew Hall	HDU, iPad displays	Tietronix @ NASA JSC
Dennis Lawler	HDU Avionics	NASA JSC
Thomas Matthews	HDU, test result vectors	NASA JSC
Arthur Molin	DC, test result vectors	Tietronix @ NASA JSC
Kristina Rojdev	HDU, anomaly detection	Tietronix @ NASA JSC
Craig Russell	HDU, anomaly detection data	Tietronix @ NASA JSC
Lui Wang	DC lead	NASA JSC

TEAMS Team	ACAWS Role	Affiliation
Somnath Deb	TEAMS expert	QSI
Sudipto Ghoshal	TEAMS expert	QSI
Deepak Haste	TEAMS expert	QSI
Venkat Malepati	TEAMS expert	QSI

Appendix B: Acronym List

ACAWS	Advanced Caution and Warning System
AMISS	Anomaly Monitoring Inductive Software System (aka IMS outside JSC)
API	Application Programming Interface
ARC	NASA Ames Research Center
C&W	Caution and Warning
C/E	Controller / Evaluator
CSV	Comma Separated Values (common file format)
CxPASS	Constellation Procedures Application Software Suite
DFT	Design for Testability
DTO	Development Test Objective
ELOG	Event Logger
EVA	Extra Vehicular Activity
FCT	Flight Control Team
FFA	Functional Fault Analysis (Ares I TEAMS modeling and analysis effort)
FN	Flight Note
FTT	Full-task Trainer
GMT	Greenwich Mean Time (aka UTC)
IMS	Inductive Monitoring System (aka AMISS at JSC)
ISP	Information Sharing Protocol
ISS	International Space Station
JSC	NASA Johnson Space Center
KSC	NASA Kennedy Space Center
LRU	Line Replaceable Unit
MCC	Mission Control Center (JSC)
MCT	Mission Control Technologies
MER	Mission Evaluation Room
MET	Mission Elapsed Time
MOD	Mission Operations Directorate (JSC)
O/E	Operator / Evaluator
PDU	Power Distribution Unit
PRACA	Problem Reporting and Corrective Action
PTT	Part-task Trainer
QSI	Qualtech Systems Inc.
RIU	Remote Interface Unit
SITF	Source-Independent Telemetry File
TEAMS	Testability Engineering and Maintenance System
Unique-identifier	PUI, MSID, CUI, etc.; a method to associate a parameter/measurement with a unique name
VV&A	Verification, Validation, and Accreditation
WSN	Wireless Sensor Node

Appendix C: Diagnostic and Repair Procedures

The ACAWS project scenarios for the Desert RaTS exercise will use a set of procedures related to diagnosis of failure. Procedures include manual diagnostic tests that are needed to resolve diagnostic ambiguity and procedures for recovery of function once the necessary recovery actions are known, based on accurate diagnosis of failure.

The first section – Scenario Procedures – describes the procedures for three defined scenarios. The scenarios are organized around closely related groups of failures, some with ambiguity when using only automated testing that is resolved with application of manual tests.

The second section – Manual Diagnostic Test Procedures – lists a procedure that corresponds to each of the manual tests in the TEAMS diagnostic model. The Scenario Section refers to these manual tests as necessary to complete the scenario procedures. The procedure names in this section correspond to the TEAMS test names.

Scenario Procedures

Three scenarios are defined:

- Scenario 1 – WSN Mote Failure. The failure is unambiguous and includes only a repair procedure
- Scenario 2 – Loss of cRIO Functionality. Failure of a power supply or the cRIO box could cause the failure, and manual diagnostic steps are needed to resolve ambiguity.
- Scenario 3 – Loss of 28 VDC equipment. The 28 VDC converter or a control card in the cRIO could cause the observable effects. Manual steps are required, and multiple repair procedures are included.

Scenario 1 – WSN Mote Failure

The scenario starts with a failure of a single Wireless Sensor Node (WSN). All sensors on the node stop reporting data, and software sets 'invalid' status on the telemetry. The diagnostic reasoner identifies the WSN node that has failed, and the failure is indicated on ACAWS displays.

Repair Procedure

1. Determine if the sensors on the WSN mote are required for continued operation
    - If YES, proceed to following steps
    - If NO, stop.
2. Locate the failed node
3. Turn the WSN Power Switch to OFF (middle position).
4. Unplug the power cord from the WSN.
5. Unplug the sensor terminals from the WSN. Check that the sensor terminals are labeled so they can be connected correctly on the new WSN. If labels are not present, apply tape to the sensor terminal wire and label it with the port number from which it was removed.
6. Pull the WSN off the mounting bracket. They are held in place via Velcro.

7. Obtain a replacement WSN from inventory.
    - A WSN can be replaced by a functionally equivalent node from a different manufacturer; a Nivis WSN could be replaced by a Dust WSN if necessary
8. Place Velcro on the new WSN in the same location as the Velcro on the mounting bracket.
9. Place the new WSN on the mounting bracket.
10. Plug the sensor terminals into the new WSN. Verify that the sensors are connected to the correct channels by checking the channel # and sensor name in the channelization table with what's plugged in.
11. Update the channelization/calibration tables so that measurements from sensors are correctly associated with and calibrated for the new node (specific steps are TBD)
12. Plug in the power cord into the WSN.
13. Flip the power switch to 28 VDC
14. Wait for the node to join the network (~ 5 – 10 minutes) and verify that sensor data is being received by checking the data displays on the iPad and observing that correct sensor data is being received and no invalid data indications are present
15. Checkout – confirm that the sensors are working
    - Use the ACAWS display and confirm that the previously failed WSN mote is not listed as a BAD or SUSPECT component and the ACAWS connectivity display shows the WSN with a white outline. Check the ACAWS WSN display and confirm that all sensor values are displayed with numeric values. Any values displayed in magenta or with an 'M' instead of a numeric value indicate invalid data. If all of the sensors are magenta or 'M' it indicates the WSN mote is not functioning. If some are displaying valid data in white text but one or more are in magenta (with 'M') it indicates that the indicated sensors were not plugged into the WSN mote correctly, or that the sensor itself has failed. In this case the ACAWS display should also list the affected sensors as BAD components.

Scenario 2 – Loss of cRIO Functionality

This scenario results in loss of cRIO functionality, either due to a 24 VDC Power Supply failure or a complete failure of the cRIO itself. The failure results in loss of power to WSN motes 1 – 11 and loss of all telemetry from the sensors on these nodes. In addition, six CO_2 sensors and 2 O_2 sensors will stop reporting data. Validity status telemetry will indicate all of these sensors as 'invalid'. Software control of Solid State Light Modules (SSLM) is lost also. Lights can still be controlled by manual switches on the lights, and lights remain in their current state at the time of the failure.

The procedure must first resolve the ambiguity of the failure. Both the 24 VDC Power Supply and a failure of the cRIO appear identical to the automated diagnostic reasoner. The

failure is similar to a 28 VDC failure, the only difference is that with a 28 VDC failure there should be valid readings on two O_2 sensors that the cRIO reads. TEAMS-RT diagnosis will not differentiate between these three failures, generating an ambiguity group consisting of the possible causes, and also listing many of the impacted components in a large SUSPECT set. If the 24 VDC Power Supply is receiving power, but is not outputting 24 VDC power, then it can be concluded that the Power Supply is the failed component and the cRIO functionality is lost as an impact of the Power Supply. If the Power Supply can be shown to be generating power, then it can be concluded that the cRIO has failed internally. This procedure assumes that the PDU port is functioning and providing power to the power supply.

Diagnostic Procedures

1. Perform the 24VDC-PowerSupply-LED-Check procedure.
    - A FAIL result indicates that the Power Supply is bad. The operator will be directed to proceed with Repair Procedure – 24 VDC Power Supply.
2. If the previous result is PASS, perform the cRIO-LED-Check procedure.
    - A FAIL result indicates that the cRIO box is bad. The operator will be directed to proceed with Repair Procedure – cRIO

Repair Procedure – 24 VDC Power Supply

1. Lift the floorboards in Segment A and locate the 24 VDC Power Supply
2. Command power to PDU B1 Port 9 to OFF.
    - Assures that no voltage surges occur when removing or installing the new power supply.
3. Unplug the power supply from the PDU.
4. Disconnect the cRIO power input from the power supply.
5. Remove the mounting bolts from the 24 VDC power supply and remove the box.
6. Obtain a replacement 24 VDC power supply from inventory.
7. Install the new power supply.
8. Connect power lines to the cRIO.
9. Plug the 24 VDC Power Supply in to the PDU Port 9.
10. Command power to PDU B1 Port 9 ON.
11. Wait for the cRIO to power up.
12. Command each WSN to ON
13. Wait for WSN nodes to join the network – 5 – 10 minutes
14. Conduct checkout of cRIO functionality – WSN data is flowing, CO_2 and O_2 sensor data is flowing, and SSLM control via SW is available.

    - Use the ACAWS display and confirm that the cRIO, 24 VDC Power Supply, and WSN motes are not listed as a BAD or SUSPECT components. The ACAWS connectivity display shows the components with white outlines. Any of these indications confirm that the 24 VDC Power Supply is providing power to the

cRIO. Problems with any WSN motes indicated on the ACAWS connectivity display or the ACAWS WSN Display could indicate that the cRIO did not boot up properly, and may require recycling power to the cRIO, repeating steps 12 – 14.

Repair Procedure – cRIO

1. Lift the floorboards in Segment A and locate the cRIO.
2. Command power to PDU B1 Port 9 to OFF.
    - This cuts power from the 24 VDC power supply to ensure that power is not hot to the cRIO.
3. Command power to the PDU B1 Port 10 to OFF
    - This removes power to the 28 VDC power supply to assure that no current is on when working on the cRIO
    - This step will remove power to external spotlights; if they are in use, consider making the repair at a time when they are not needed, but if not practical, this step can be skipped.
4. Disconnect the cRIO power input from the 24 VDC Power Supply.
5. Disconnect WSN power control connector that runs to the WSN Interface box
    - The connector includes 11 WSN neutral power lines and 6 CO_2 neutral power lines.
6. Disconnect the 6 CO_2 sensor inputs and 2 O_2 sensor inputs. Make sure that the connectors are labeled so that they can be connected in the same way to the replacement cRIO.
7. Disconnect the SSLM Lighting Control connector that runs to the Lighting Interface Box
8. Disconnect 8 HIMS sensor inputs from the cRIO. Assure they are labeled so they can be reconnected in the same location to the replacement cRIO.
9. Disconnect the Ethernet connector from the cRIO.
10. Remove the mounting fasteners from the cRIO and remove the box.
11. Obtain a replacement cRIO from inventory.
12. Install the new cRIO.
13. Connect the Ethernet connector to the cRIO.
14. Connect 8 HIMS sensor input lines.
15. Connect SSLM Lighting Control connector.
16. Connect the 6 CO_2 and 2 O_2 sensors to the same ports as they were on the old cRIO.
17. Connect the WSN and CO_2 power control connector to the WSN Interface Box.
18. Connect the power input to the 24 VDC Power Supply.
19. Command power to PDU Port 10 to ON to supply 28 VDC power (if turned off at step 3).
20. Command power to PDU Port 9 to ON to supply 24 VDC power to the cRIO.

21. Wait for the cRIO to power up.
22. Command each WSN to ON
23. Wait for WSN nodes to join the network – 5 – 10 minutes
24. Conduct checkout of cRIO functionality
    - Check that one or more sensor values are reported from each of 11 WSN motes. Use the ACAWS displays to observe that sensor data is reported as valid and the data is updating. This verifies that the connections to the WSN control lines are all correct.
    - Check that data from each of 6 CO_2 and 2 O_2 sensors is available. Refer to ACAWS displays to observe that CO_2 and O_2 sensor data is reported as valid and data is updating. This confirms that power control connectors to the CO_2 sensors are correct, and that data input connections from CO_2 and O_2 sensors are also correct, and that the Ethernet is connected correctly and the cRIO is communicating.
    - Send a command to one of the SSLMs to turn the light OFF and back ON (or ON, then OFF). Commands can be sent from any location with commanding authority - iPad display, MCC, procedure execution tools or possibly other applications. Successful command execution verifies the Lighting Control interface was connected correctly.
    - Check HIMS data. Use HIMS procedures/documentation for this. This verifies that HIMS data interfaces were connected correctly. If HIMS is off at the time of the test, execute HIMS Initialization Procedures prior to the test.

Scenario 3 – Loss of 28 VDC Power

This scenario involves loss of 28 VDC power to the WSN motes and CO_2 sensors. It could be caused by a loss of 28 VDC converter but the cRIO 9477 card failure has nearly identical impacts. A manual test is needed to differentiate. One approach is to turn on the exterior spotlights and see if they turn on. Another is to perform troubleshooting under the floor to determine if power is being generated.

It is also possible that a 9477 Card Failure could result in partial loss, either with some but not all WSN or CO_2 sensors losing power, or intermittent power to some or all WSN and CO_2 sensors. If this occurs and the WSN or sensor can be confirmed to be good (connecting to an unaffected power source, or other testing), the cRIO or the 9477 Card will need to be replaced.

If a replacement cRIO 9477 Card is available, it can be replaced easily without needing to power down the cRIO. It may also be possible to swap cRIO 2 for cRIO 1, since there is much less equipment on cRIO 2 in the X-Hab unit.

Diagnostic Procedures

1. Perform the Spotlight-Check procedure on any one of the spotlights (B, D, F, H)

- The operator should check the ACAWS Diagnostic Display for diagnostic reasoner conclusions based on the results of the Spotlight-Check procedure.
- Test Passes – will be directed to execute Repair Procedure - cRIO 9477 Card Replacement
- Test Fails – will be directed to execute Repair Procedure – 28 VDC Replacement
- Test Unknown – will be directed to execute additional electrical system trouble-shooting procedures

2. If results from the Spotlight-Check are UNKNOWN, perform the *28VDC-Converter-Power-Check* procedure.

Repair Procedure – 28 VDC Power Supply

1. Lift the floorboards in Segment B and locate the 28 VDC Converter.
2. Command power to PDU B1 Port 10 to OFF.
    - This cuts power to the 28 VDC Converter to ensure that power is not hot during the procedure.
3. Disconnect power input from the PDU.
4. Disconnect power output to the Terminal Strip Box.
5. Remove mounting fasteners from the 28 VDC Converter and remove the converter.
6. Obtain a replacement 28 VDC Converter from inventory.
7. Install the 28 VDC Converter using mounting hardware from the failed unit.
8. Connect power outputs to the Terminal Strip Box.
9. Connect power to the PDU.
10. Command PDU B1 Port 10 to ON using any suitable commanding method – iPad displays, procedure tools, MCC or other.
11. If necessary, command the WSN motes and CO_2 sensor power to ON
    - Depends on how the cRIO switches default when it powers up; if they default to OFF they will need to be commanded ON, otherwise this step is not needed
12. Wait for the WSN motes to join the network – 5-10 minutes
13. Verify 28 VDC power by observing any powered item – any CO_2 sensor, WSN mote or external spotlight confirm that 28 VDC power is available. ACAWS diagnosis can verify repair success; it will report that 28 VDC is good (by removing it from the BAD and SUSPECT list) as soon as any 28 VDC equipment tests pass; intermediate diagnoses may list various SUSPECT components until all WSN motes have joined the network. When all 28 VDC equipment is fully functional, ACAWS will report nominal conditions with no BAD or SUSPECT components related to 28 VDC power.

Repair Procedure – cRIO 9477 Card

1. Lift the floorboard in Segment A and locate the cRIO.
2. Locate the 9477 Card slot in the cRIO. Remove the connector.

3. Remove the 9477 Card from the slot.
4. Obtain a replacement card from inventory.
5. Install the 9477 Card in the slot.
6. Connect the multi-pin connector to the card.
7. If necessary, command the WSN motes and CO_2 sensor power to ON
    - Depends on how the cRIO switches default when it powers up; if they default to OFF they will need to be commanded ON, otherwise this step is not needed
8. Wait for the WSN motes to join the network – 5-10 minutes
9. Conduct checkout of the WSN channels to assure they have powered up correctly
    - Use the ACAWS display to check that WSN motes are indicated with a white outline, indicating that the WSN is working. If some but not all WSN motes are functioning, there may be a problem with a connector or possible a command channel was not set. Initially try commanding the channel to OFF and then back ON, wait for the power-up time and check again. If the WSN mote still does not come up, there may be a problem with the connector; remove the connector, carefully inspect the pins on the connector and if there does not seem to be any bent or damaged pins, reconnect and try again.

Manual Diagnostic Test Procedures

There are several tests in the TEAMS Model that rely on manual tests to be performed. Procedures for executing these tests are outlined here. Some of these are simple one-step actions; some are basic outlines of more complicated activities. In all cases the operator is expected to indicate a test pass or failure via the Procedure Display tool. The procedure names used are the TEAMS test name with which each procedure is associated.

24VDC-PowerSupply-LED-Check

1. Lift the floorboard in Segment A where the cRIO and 24 VDC Power Supply are located.
2. Locate the 24 VDC Power Supply and examine the box. Look for a green LED indicator light. If the light is out, report 'LED Test Fail' in the procedure display.
3.

CabinTempTest

1. This is an operator check of cabin temperature based on an operator's subjective assessment of the temperature. If the crewmember feels that the temperature is significantly warmer or colder than temperature settings, report FAIL, otherwise PASS.

HabitatTempTest

1. This is an operator check of the habitat module (X-Hab) based on an operator's subjective assessment of the temperature. If the crewmember feels that the

temperature in the inflatable habitat module is significantly warmer or colder than temperature settings, report FAIL, otherwise PASS.

One-Light-Off

1. This test would normally be done only when a problem with lighting is evident, such as a light burning out or being unresponsive to either commands or switch activation. Crewmembers would not normally be asked to methodically check all lights using this procedure.
2. If a single light goes out or can't be turned on either by commanding with crew user interface or with manual switches on the light module, but other lights function normally, report FAIL. If all lights are in their expected state report PASS.

Power-to-SSLMX-ok

1. If the light is on when commanded or when turned on using manual switches, report PASS. Otherwise report FAIL.

   NOTE – there are TEAMS tests for each of 8 Solid State Light Modules (SSLMs). Substitute the SSLM number for 'X' to check each light individually. SSLM 1 is in Segment A, SSLM 2 in Segment B, etc.

Lights-commanded-on-ok

1. This is a test to determine if operator actions have been conducted as expected.
2. If a light is not in its expected ON/OFF state, check command logs to determine if a light command has been sent to the light in question, and it was the desired ON or OFF command. If the correct command was sent, report PASS. If an incorrect command was sent, or a desired command was not sent, report FAIL.

Light-Command-Response-Test

This test checks the lighting command pathways using any of the available SSLMs.

1. Select a light that can be turned on and off with minimal disturbance to other work.
2. If the selected SSLM is currently OFF, send an ON command; if it is currently ON, send an OFF command.
3. If the SSLM responds to the command, report PASS, otherwise FAIL.
4. Send a command to restore the light to its original state, as desired.

Manual-Switch-is-on-ok

This test is performed on a light that is not on, and commands to turn it on have not succeeded.

1. Check the SSLM Master switch position. If the switch is in the ON position report PASS, otherwise FAIL.

Dimmer-Switch-is-bright-ok

This test is performed on a light that is not on and commands to turn it on have not succeeded.

1. Check the dimmer knob position. If it is rotated fully or mostly toward the DIM position (counterclockwise) report FAIL. If it is more than three-fourths to the BRIGHT position (clockwise) report PASS. If turning the dimmer toward BRIGHT turns the light ON, report FAIL; this will let the diagnostic reasoned know that the cause of the problem was an inadvertent dimmer position. The dimmer should be set at the desired setting at the conclusion of the test.

Multiple-Lights-Out

This test is used primarily to determine if power is available to the SSLMs. No operator actions are needed; if the problem were caused by gross mismanagement of lighting procedures, other diagnostic and procedure steps will direct the operator to command and configure lights.

1. If any SSLM is ON, report PASS, otherwise report FAIL.

Trash-Compactor-Check

1. Turn the Trash Compactor ON. If it operates, report PASS. If it does not operate at all, report FAIL.

HIMS-Display-Check

This is a check for power to the HIMS display.

1. If the HIMS display is currently OFF, turn it ON.
2. If the HIMS display is working, report PASS, otherwise report FAIL.

Task-Light-Check

This is a check for power to the task light in the General Maintenance Work Area.

1. Check that the GMWS Task Light is plugged into the power strip.
2. Turn the light ON. If it turns on, report PASS, otherwise report FAIL.

GMWS-Hoist-Check

This is a check for power to the GMWS Hoist.

1. Locate the GMWS Hoist controls in Segment D.
2. If the GMWS Hoist is not currently in use, check the area to make sure that brief activation of the hoist will not cause damage or injury. If the hoist is in use, coordinate with the operator using it to conduct the test.
3. Briefly activate and then immediately deactivate the hoist. If it operates, report PASS. If it does not operate at all, report FAIL.

GMWS-PowerStrip1-Check

1. Plug any available free-standing electrical equipment into GMWS Power Strip # 1 in Segment D. The vacuum cleaner, a light, cell phone charger or anything that provides an obvious response when turned on, will suffice. If the item operates when turned ON, report PASS, otherwise report FAIL.

Spotlight-Check

Any one of four TEAMS tests can be used: Spot-B-Check, Spot-D-Check, Spot-F-Check, Spot-H-Check.

These checks determine if external spotlights are functioning. They require an observation of the lights by a person. Spotlight F may be visible from the exterior camera, the others require an exterior crewmember during the day. At night it may be possible to tell if a light has come on by looking out a window or using the exterior camera.

1. If there is an exterior crew member in the vicinity of the HDU, determine the crewmember's location to determine which light is visible from the location. Turn on the selected light using the wall switch located near the Segment A hatch.
2. If an external crewmember is not available to assist with the test, use the external camera to see if the light comes on. Slew the exterior camera to point at spotlight H using the camera controls at the GeoLab. Turn on Spotlight H with the wall switch located near the Segment A hatch.
3. Determine if the light turned on, either from exterior crewmember report or from observing the light with the camera. If the light is on, report PASS, and if the light is not on, report FAIL. If the light can't be seen with the camera due to sunlight or slewing constraints, report UNKNOWN.

Internal-Outlet-Check

1. Plug in a free-standing device to the internal outlet located in Segment D. Any 120 VAC equipment with a standard plug will work.
2. Turn the item on. If it runs normally, report PASS, if it does not operate, report FAIL.

External-Power-Switch-is-not-off

This test is used to determine if the power was inadvertently turned off from the main power switch.

1. If there is any power to any equipment in the HDU, report PASS.
2. If there is no power in the HDU, check the main power switch. If the switch is OFF, report FAIL. If the switch is ON report PASS.

WSN-Signal-Check

This is a test to determine if the cRIO 9477 Card is successfully controlling power to the WSN Interface Box. There is currently not a tester that could perform this test, and the

simplest means to perform this check would be to replace the cRIO 9477 Card and see if the new one works.

1. Execute Repair Procedure - cRIO 9477 Card.
2. If the new cRIO Card works correctly, report PASS. If the card does not work, report FAIL.
3. If the test fails, replace the original cRIO 9477 Card in the cRIO and return the new one to inventory.

cRIO-LED-Check

1. Lift the floorboard opening in Segment A and locate the cRIO.
2. Check for a green LED on the box. If the light is ON, report PASS. If no light is observed, report FAIL.

WSN-IF-Continuity-Check

1. Lift the floorboard opening in Segment B and locate the WSN Interface Box.
2. Remove the cover from the WSN Interface Box.
3. Use an electrical test tool to conduct a conductivity test by placing one electrode of the tester on an available test point at the input side of the box, and one electrode on an exposed wiring connector at the output side of the box.
4. If the electrical tester indicates that there is an electrical path through the box, report PASS. Otherwise, report FAIL.
5. Replace the cover on the WSN Interface Box.

28VDC-Converter-Power-Check

1. Lift the floorboard in Segment B where the 28 VDC is located. Locate the 28 VDC Converter unit.
2. Inspect the 28 VDC converter for a red indicator light. If the light is not on, report FAIL.
    - The converter is probably producing power if the indicator light is on.
3. Confirm power generation by using an electrical tester to read the current from the 28 VDC Converter. Locate available current test points and at the converter output or in the WSN Interface Box and contact the tester leads to read voltage output. If the current reading is above 26 volts, report PASS, otherwise report FAIL.
4. If tests can't be completed, report UNKNOWN.

Appendix D: TEAMS Diagnostic Model

The TEAMS model developed from the schematics of the HDU, is partitioned hierarchically into seven subsystems: Power, Avionics, Communications, Geolab, Environmental Control and Monitoring, Crew Accommodations and Food Production.

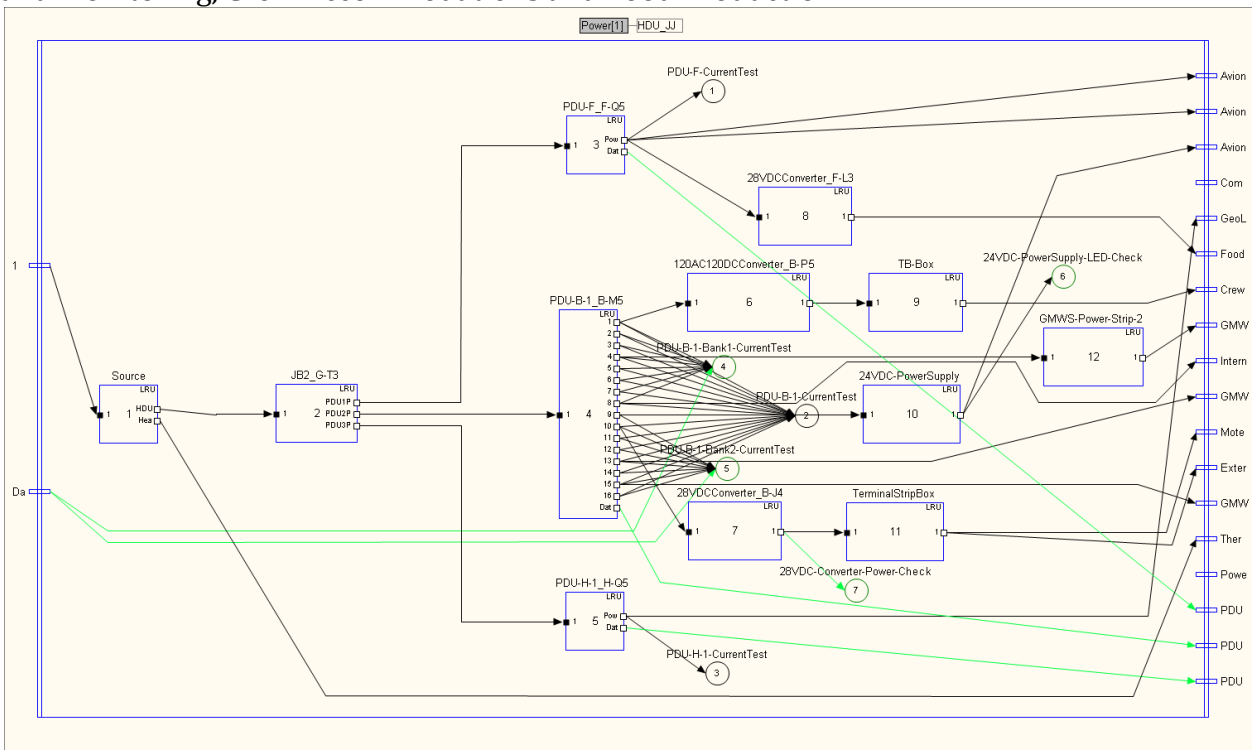

Figure 13: Power Subsystem

TEAMS Module	Failure Mode	Failure Signals
120AC120DCConverter_B-P5	ACDCFailure	120DC-Power ACDCFailure, Data
24VDC-PowerSupply	24VDC-PowerSupply-failure	24VDC-Power 24VDC-Power2 light-command
28VDCConverter_B-J4	28VDC-B-Failure	B1-28VDC-Power
28VDCConverter_F-L3	28VDC-F-Failure	F-converter
GMWS-Power-Strip-2	GMWS-Power-Strip-2-failure	B1-Power
JB2_G-T3	JunctionBoxFailure	24VDC-Power 24VDC-Power2 B1-28VDC-Power B1-Power F2-Power H1-Power JunctionBoxFailure Power
PDU-B-1_B-M5	PDU-B-1-Bank1CurrentSensorFailure	PDU2Bank1CurrentSensorFailure
	PDU-B-1-No-Current-Failure	24VDC-Power

		24VDC-Power2 B1-28VDC-Power B1-Power Data light-command PDU2Failure Power
	PDU-B-1-Bank2CurrentSensorFailure	PDU2Bank2CurrentSensorFailure
	PDU-B-1-CurrentSensorFailure	PDU2CurrentSensorFailure
	PDU-B-1-Total-Failure	24VDC-Power 24VDC-Power2 B1-28VDC-Power B1-Power Data light-command PDU2Failure Power Total-Power-Failure
PDU-F_F-Q5	PDU-F-DataFailure	Data
	PDU-F-Failure	Data F2-Power PDUFailure
PDU-H-1_H-Q5	PDU-H-1-Failure	Data H1-Power Power
	PDU-H-1-DataFailure	Data
Source	SourceFailure	24VDC-Power 24VDC-Power2 B1-28VDC-Power B1-Power F2-Power H1-Power Power SourceFailure
TB-Box		
TerminalStripBox		
120AC120DCConverter_B-P5	ACDCFailure	120DC-Power ACDCFailure Data

TEAMS Testpoint	TEAMS Test	Failure Signals
PDU-F-CurrentTestpoint	PDU-F-CurrentTest	Power, F2-Power
PDU-B-1-CurrentTestpoint	PDU-B-1-CurrentTest	Power, PDU2Failure, Power
	PDU-B-1-CurrentValidityTest	Total-Power-Failure, PDU2CurrentSensorFailure
PDU-H-1-CurrentTestpoint	PDU-H1-CurrentTest	Power, H1-Power
PDU-B-1-Bank1-CurrentTestpoint	PDU-B-1-Bank1-CurrentTest	PDU2Failure, Power
	PDU-B-1-Bank1-CurrentValidityTest	PDU2Bank1CurrentSensorFailure, Total-Power-Failure
PDU-B-1-Bank2-CurrentTestpoint	PDU-B-1-Bank2-CurrentTest	PDU2Failure,

		Power
	PDU-B-1-Bank2-CurrentValidityTest	PDU2Bank2CurrentSensorFailure, Total-Power-Failure
24VDC-PowerSupply-LED-CheckTestpoint	24VDC-PowerSupply-LED-Check	24VDC-Power
28VDC-Converter-Power-Check	28VDC-Converter-Power-Check	B1-28VDC-Power

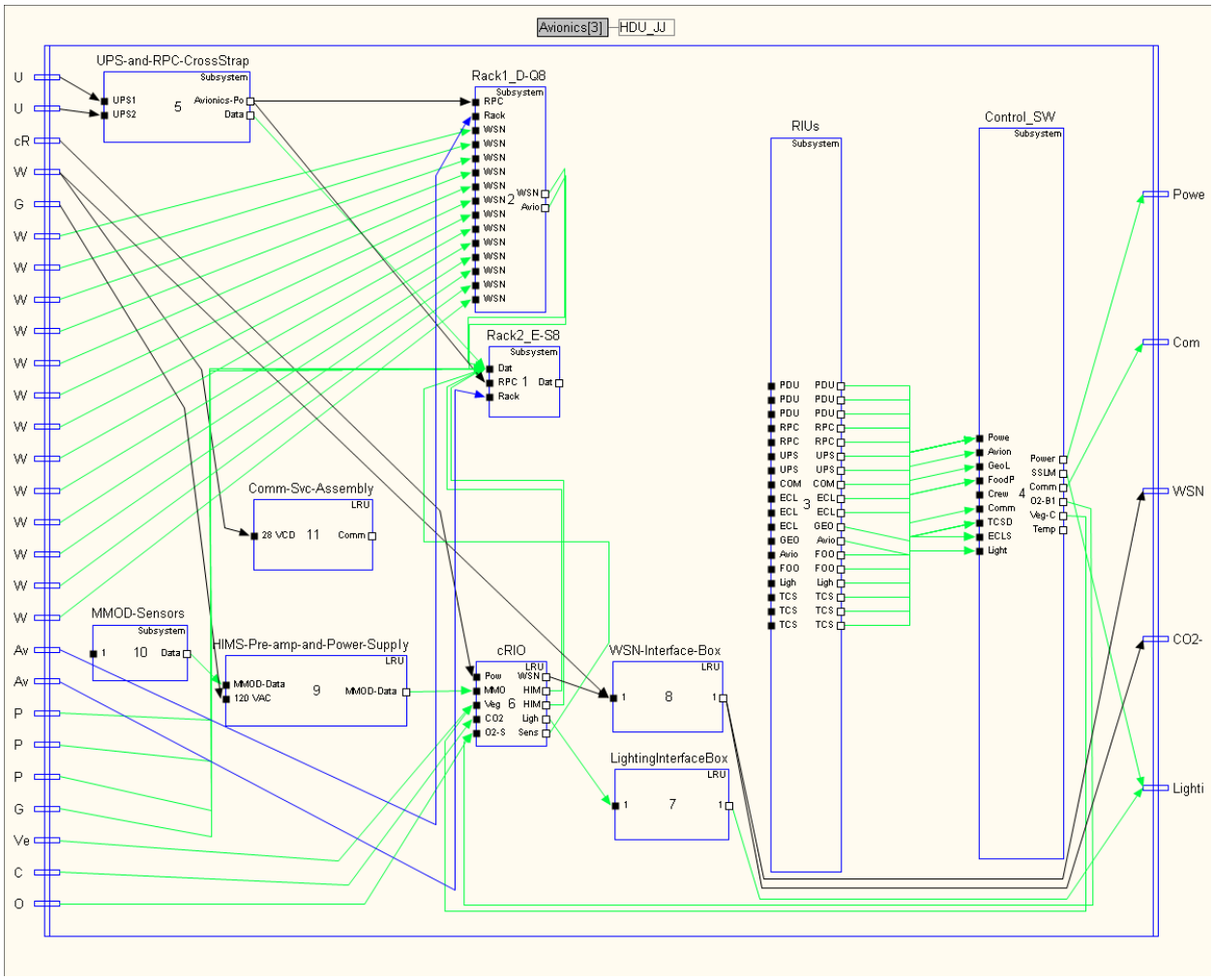

Figure 14: Avionics Subsystem

TEAMS Module	Failure Mode	Failure Signals
AVDAQ_D-N3	**AVDAQPowerFailure**	**AVDAQFailure** **Data** **Power**
	AVDAQDataFailure	**AVDAQFailure** **Data**
AVSwitch_D-L7	**AVSwitchDataFailure**	**Data**
CDHSwitch_E-Q7	**CDHSwitchFailure**	**CDHSwitchFailure** **Data**
	CDHSwitchPowerFailure	**Data** **Power**
Comm-Svc-Assembly	**CSA-Failure**	**ExternalCommunications**

Control_SW	**ECLSSControlSWFailure**	**ECLSSControlSWFailure** **O2-Data**
	LightingControlSWFailure	**light-command** **LightingControlSWFailure**
	PowerControlSWFailure	**PowerControlSWFailure**
	CommsControlSWFailure	**CommsControlSWFailure** **Wireless-Data**
	FoodControlSWFailure	**FoodControlSWFailure**
	GeoLabControlSWFailure	**GeoLabControlSWFailure**
	CrewAccomControlSWFailure	**CrewAccomControlSWFailure**
	AvionicsControlSWFailure	**AvionicsControlSWFailure**
	TCSControlSWFailure	**TCSControlSWFailure**
cRIO	**cRIO-failure**	**24VDC-Power** **24VDC-Power2** **cRIO-failure** **Data** **light-command**
	9223-HIMS-Preamp-Data2-failure	**Data**
	9477-WSN-control-failure	**B1-28VDC-Power** **cRIO-28VDC-Ctrl**
	9223-HIMS-Preamp-Data1-failure	**Data**
	9219-SensorData-failure	**cRIO-DAQ**
	9476-LightingInterfaceBoxControl-failure	**light-command**
DataServer_E-O2	**DataServerFailure**	**Data** **DataServerFailure** **Power**
DECTVOIP_D-N4	**DECTVOIPDataFailure**	**Data** **DECTVOIPFailure**
	DECTVOIPPowerFailure	**Data** **DECTVOIPFailure** **Power**
HIMS-Pre-amp-and-Power-Supply	**HIMS-Pre-amp-and-Power-Supply-failure**	**Data** **Power**
LightingInterfaceBox	**LightingInterfaceBox-failure**	**light-command**
MMOD-Sensors		
MMOD-1-Sensor	**MMOD-1-Sensor-failure**	**MMOD-1-failure**
MMOD-2-Sensor	**MMOD-2-Sensor-failure**	**MMOD-2-failure**
MMOD-3-Sensor	**MMOD-3-Sensor-failure**	**MMOD-3-failure**
MMOD-4-Sensor	**MMOD-4-Sensor-failure**	**MMOD-4-failure**
MMOD-5-Sensor	**MMOD-5-Sensor-failure**	**MMOD-5-failure**
MMOD-6-Sensor	**MMOD-6-Sensor-failure**	**MMOD-6-failure**
MMOD-7-Sensor	**MMOD-7-Sensor-failure**	**MMOD-7-failure**
MMOD-8-Sensor	**MMOD-8-Sensor-failure**	**MMOD-8-failure**
NAT_D-N3	**NATPowerFailure**	**Data** **NATFailure** **Power**
	NATDataFailure	**Data** **NATFailure**

NVR_D-N5	**NVRPowerFailure**	**Data NVRFailure Power**
	NVRDataFailure	**Data NVRFailure**
OC1_E-P4	**OC1Failure**	**Data OC1Failure Power**
OC2_E-P5	**OC2Failure**	**Data OC2Failure Power**
Rack1_D-Q8		
Rack2_E-S8		
RIUs	**CommsRIUFailure**	**CommsRIUFailure Wireless-Data**
	RPC2RIUFailure	**RPC2RIUFailure**
	ECLSSRIU1Failure	**ECLSSRIU1Failure O2-Data**
	RPC1RIUFailure	**RPC1RIUFailure**
	UPS2RIUFailure	**UPS2RIUFailure**
	AvionicsRacksRIUFailure	**AvionicsRIUFailure**
	ECLSSRIU2failure	**ECLSSRIU2failure**
	TCSRIU3Failure	**TCSRIU3Failure**
	UPS1RIUFAilure	**UPS1RIUFAilure**
	LightingRIUFailure	**light-command LightingRIUFailure**
	TCSRIU1Failure	**TCSRIU1Failure**
	TCSRIU2Failure	**TCSRIU2Failure**
	PDURIU3Failure	**PDU3RIUFailure**
	FoodProdRIU2Failure	**FoodProdRIUFailure**
	FoodProdRIU1Failure	**FoodProdRIUFailure Veg-CO2-Data**
	PDURIU2Failure	**PDU2RIUFailure**
	GeoLabRIUFailure	**GeoLabRIUFailure**
	PDURIU1Failure	**PDU1RIUFailure**
RPC1_D-R7	**RPCPowerFailure**	**Data Power**
	RPC-Current-Sensor-Failure	**Data RPC1CurrentSensorFailure**
RPC2_D-R6	**RPC2-Current-Sensor-Failure**	**Data RPC2CurrentSensorFailure**
	RPC2PowerFailure	**Data Power RPC2PowerFailure**
UPS1_D-T7	**UPS1PowerFailure**	**Data Power**
	UPS1-Current-Sensor-Failure	**Data UPS1CurrentSensorFailure**
UPS2_D-T6	**UPS2-Current-Sensor-Failure**	**Data UPS2CurrentSensorFailure**

	UPS2PowerFailure	**Data** **Power**
UPS-and-RPC-CrossStrap		
WNC_D-N7	**WNCDataFailure**	**Data** **WNCDataFailure**
	WNCPowerFailure	**Data** **Power** **WNCDataFailure**
WSNGateway_D-N4	**WSNGatewayFailure**	**Data** **Power**
WSN-Interface-Box	**WSN-Interface-Box-failure**	**B1-28VDC-Power** **WSN-IF-Continuity**

TEAMS Testpoint	TEAMS Test	Failure Signals
Airlock-CO2-Sensor Testpoint	Airlock-CO2-Validity	24VDC-Power, B1-Power, 24VDC-Power
	Airlock-CO2-Sensor	Airlock-CO2, cRIO-DAQ, B1-28VDC-Power
AVDAQTestpoint	AVDAQ-Heartbeat-Test	AVDAQFailure, F2-Power
AvionicsControlSWTestpoint	AvionicsControlSW-Heartbeat-Test	AvionicsControlSWFailure
AvionicsRacksRIU1Testpoint	AvionicsRacksRIU1-Heartbeat-Test	AvionicsRIUFailure
AVSwith-Testpoint	AVSwith-Test	Data
CDHSwitchTestpoint	CDHSwitch-Heartbeat-Test	CDHSwitchFailure, F2-Power
CommsControlSWTestpoint	CommsControlSW-Heartbeat-Test	CommsControlSWFailure
CommsRIU1Testpoint	CommsRIU1-Heartbeat-Test	CommsRIUFailure
CrewAccomSWTestpoint	CrewAccomSW-Heartbeat-Test	CrewAccomControlSWFailure
cRIO-Heartbeat-Testpoint	cRIO-Heartbeat-Test	cRIO-failure, 24VDC-Power
cRIO-LED-Check	cRIO-LED-Check	24VDC-Power
DataServerFailure	DataServer-Heartbeat-Failure	DataServerFailure, F2-Power
DECTVOIPTest	DECTVOIP-Heartbeat-Test	DECTVOIPFailure F2-Power
ECLSSControlSWTest	ECLSSControlSW-Heartbeat-Test	ECLSSControlSWFailure
ECLSSRIU1Test	ECLSSRIU1-Heartbeat-Test	ECLSSRIU1Failure
ECLSSRIU2Test	ECLSSRIU2-Heartbeat-Test	ECLSSRIU2failure
External-Comm-Check	External-Comm-Check	ExternalCommunications, B1-28VDC-Power
FoodControlSWTest	FoodControlSW-Heartbeat-Test	FoodControlSWFailure
FoodProdRIU1Test	FoodProdRIU1-Heartbeat-Test	FoodProdRIUFailure
FoodProdRIU2Test	FoodProdRIU2-Heartbeat-Test	FoodProdRIUFailure
GeoLabControlSWTest	GeoLabControlSW-Heartbeat-Test	GeoLabControlSWFailure
GeoLabRIU1Test	GeoLabRIU1-Heartbeat-Test	GeoLabRIUFailure
Hygiene-CO2-SensorTestpoint	Hygiene-CO2-Validity-Test	24VDC-Power, B1-Power, cRIO-failure
Hygiene-CO2-Sensor-Testpoint	Hygiene-CO2-Sensor-Test	Hygiene-CO2, cRIO-DAQ, B1-28VDC-Power

LightingControlSWTest	LightingControlSW-Heartbeat-Test	LightingControlSWFailure
MMOD-1-SensorTestpoint	MMOD-1-SensorTest	Data, MMOD-1-failure, 24VDC-Power, B1-Power
MMOD-2-Sensor Testpoint	MMOD-2-SensorTest	Data, MMOD-2-failure, 24VDC-Power, B1-Power
MMOD-3-Sensor Testpoint	MMOD-3-SensorTest	Data, MMOD-3-failure, 24VDC-Power, B1-Power
MMOD-4-Sensor Testpoint	MMOD-4-SensorTest	Data, MMOD-4-failure, 24VDC-Power, B1-Power
MMOD-5-Sensor Testpoint	MMOD-5-SensorTest	Data, MMOD-5-failure, 24VDC-Power, B1-Power
MMOD-6-Sensor Testpoint	MMOD-6-SensorTest	Data, MMOD-6-failure, 24VDC-Power, B1-Power
MMOD-7-Sensor Testpoint	MMOD-7-SensorTest	Data, MMOD-7-failure, 24VDC-Power, B1-Power
MMOD-8-Sensor Testpoint	MMOD-8-SensorTest	Data, MMOD-8-failure, 24VDC-Power, B1-Power
NATTestpoint	NAT-Heartbeat-Test	NATFailure, F2-Power
O2-Airlock-SensorTestpoint	O2-Airlock-Validity Test	24VDC-Power, cRIO-failure
O2-Airlock-Sensor Testpoint	O2-Airlock-Sensor Test	O2-Airlock, cRIO-DAQ, H1-Power
O2-B1-Sensor Testpoint	O2-B1-Validity Test	24VDC-Power, cRIO-failure
O2-B1-Sensor	O2-B1-Sensor	O2-B1, cRIO-DAQ, H1-Power
OC1Test	OC1-Heartbeat-Test	OC1Failure, F2-Power
OC2Test	OC2-Heartbeat-Test	OC2Failure, F2-Power
PDURIU1Test	PowerRIU1-Heartbeat-Test	PDU1RIUFailure
PDURIU2Test	PDU2RIU-Heartbeat-Test	PDU2RIUFailure
PDURIU3Test	PDU3RIU-Heartbeat-Test	PDU3RIUFailure
PowerControlSWTest	PowerControlSW-Heartbeat-Test	PowerControlSWFailure
RPC1CurrentTest	RPC1CurrentTest	F2-Power
	RPC1CurrentSensorTest	RPC1CurrentSensorFailure
RPC1RIUTest	RPC1RIU-Heartbeat-Test	RPC1RIUFailure
RPC2CurrentTest	RPC2CurrentTest	RPC2PowerFailure,

		F2-Power
	RPC2Current-Sensor-Test	RPC2CurrentSensorFailure
RPC2RIUTest	RPC2RIU-Heartbeat-Test	RPC2RIUFailure
TCSControlSWTest	TCSControlSW-Heartbeat-Test	TCSControlSWFailure
TCSRIU1Test	TCSRIU1-Heartbeat-Test	TCSRIU1Failure
TCSRIU2Test	TCSRIU2-Heartbeat-Test	TCSRIU2Failure
TCSRIU3Test	TCSRIU3-Heartbeat-Test	TCSRIU3Failure
UPS1CurrentTest	UPS1CurrentTest	F2-Power
	UPS1Current-Sensor-Test	UPS1CurrentSensorFailure
UPS1RIUTest	UPS1RIU-Heartbeat-Test	UPS1RIUFAilure
UPS2CurrentTest	UPS2CurrentTest	F2-Power
	UPS2Current-Sensor-Test	UPS1CurrentSensorFailure
UPS2RIUTest	UPS2RIU-Heartbeat-Test	UPS2RIUFailure
Veggie-CO2-A1-Sensor	Veggie-CO2-A1-Validity	24VDC-Power, B1-Power, cRIO-failure
	Veggie-CO2-A1-Sensor	Veg-CO2-A1, cRIO-DAQ, B1-28VDC-Power
Veggie-CO2-C1-Sensor	Veggie-CO2-C1-Validity	24VDC-Power, B1-Power, cRIO-failure
	Veggie-CO2-C1-Sensor	Veg-CO2-C1, cRIO-DAQ, B1-28VDC-Power
Veggie-CO2-E1-Sensor	Veggie-CO2-E1-Validity	24VDC-Power, cRIO-failure
	Veggie-CO2-E1-Sensor	Veg-CO2-E1, cRIO-DAQ, B1-28VDC-Power
Veggie-CO2-G1-Sensor	Veggie-CO2-G1-Validity	24VDC-Power, B1-Power, cRIO-failure
	Veggie-CO2-G1-Sensor	Veg-CO2-G1, cRIO-DAQ, B1-28VDC-Power
WNCTest	WNC-Heartbeat-Test	WNCDataFailure, F2-Power
WSN-IF-Continuity-Check	WSN-IF-Continuity-Check	WSN-IF-Continuity
WSN-Signal-Check	WSN-Signal-Check	B1-28VDC-Power, cRIO-28VDC-Ctrl

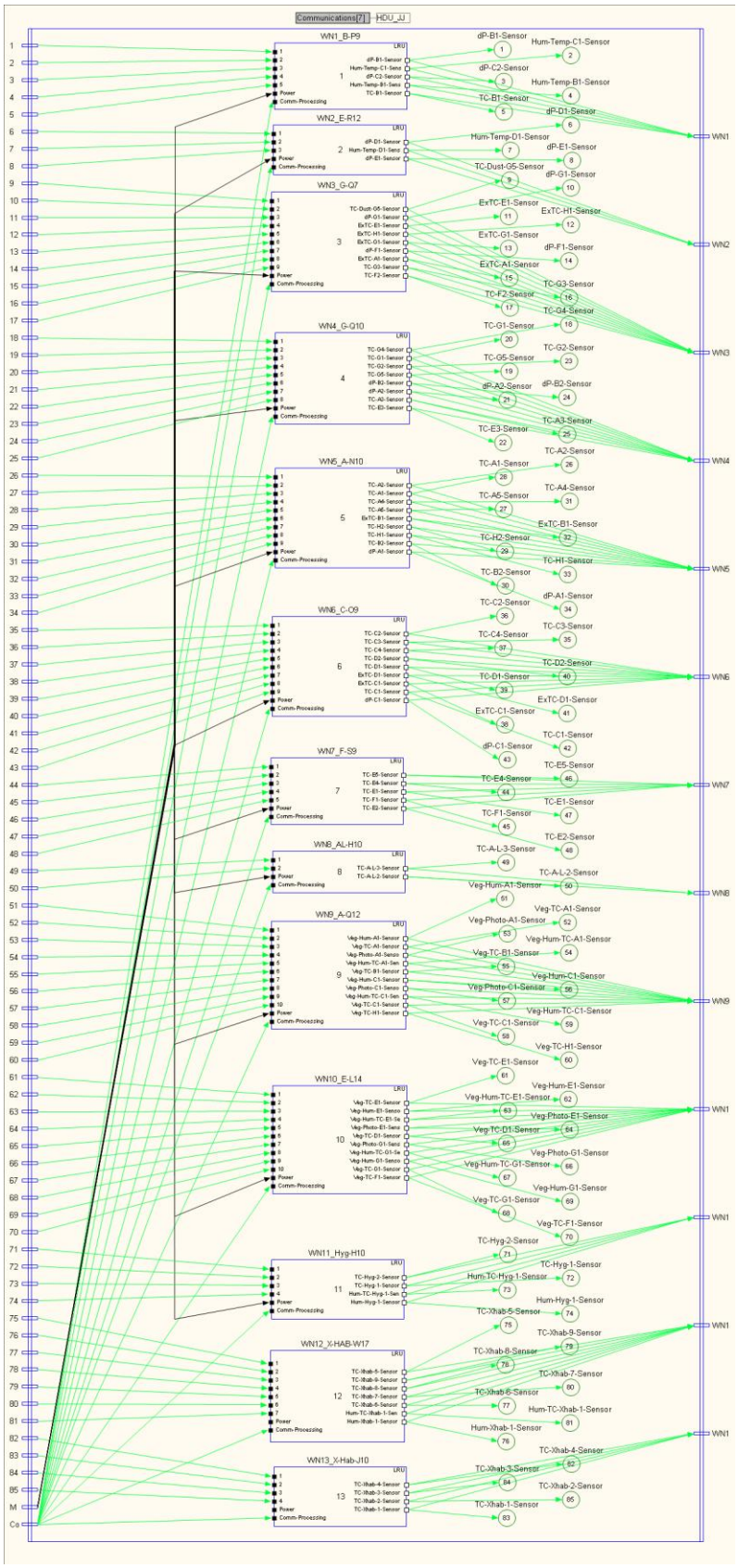

Figure 15: Communication Subsystem

TEAMS Module	Failure Mode	Failure Signals
WN1_B-P9	WN1Failure	24VDC-Power
WN10_E-L14	WN10Failure	24VDC-Power
WN11_Hyg-H10	WN11Failure	24VDC-Power
WN12_X-HAB-W17	WN12Failure	24VDC-Power
WN13_X-Hab-J10	WN13Failure	24VDC-Power
WN2_E-R12	WN2Failure	24VDC-Power
WN3_G-Q7	WN3Failure	24VDC-Power
WN4_G-Q10	WN4Failure	24VDC-Power
WN5_A-N10	WN5Failure	24VDC-Power
WN6_C-O9	WN6failure	24VDC-Power
WN7_F-S9	WN7Failure	24VDC-Power
WN8_AL-H10	WN8Failure	24VDC-Power
WN9_A-Q12	WN9Failure	24VDC-Power

TEAMS Testpoint	TEAMS Test	Failure Signals
dP-A1-Sensor	dP-A1-Sensor	dP-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-A2-Sensor	dP-A2-Sensor	dP-A2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-B1-Sensor	dP-B1-Sensor	dP-B1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-B2-Sensor	dP-B2-Sensor	dP-B2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-C1-Sensor	dP-C1-Sensor	dP-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-C2-Sensor	dP-C2-Sensor	dP-C2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-D1-Sensor	dP-D1-Sensor	dP-D1-Sensor, 24VDC-Power, B1-28VDC-Power, Wireless-Data,
dP-E1-Sensor	dP-E1-Sensor	dP-E1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-F1-Sensor	dP-F1-Sensor	dP-F1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
dP-G1-Sensor	dP-G1-Sensor	dP-G1-Sensor,

		24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-A1-Sensor	ExTC-A1-Sensor	ExTC-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-B1-Sensor	ExTC-B1-Sensor	ExTC-B1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-C1-Sensor	ExTC-C1-Sensor	ExTC-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-D1-Sensor	ExTC-D1-Sensor	ExTC-D1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-E1-Sensor	ExTC-E1-Sensor	ExTC-E1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-G1-Sensor	ExTC-G1-Sensor	ExTC-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
ExTC-H1-Sensor	ExTC-H1-Sensor	ExTC-H1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-Hyg-1-Sensor	Hum-Hyg-1-Sensor	Hum-Hyg-1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-TC-Hyg-1-Sensor	Hum-TC-Hyg-1-Sensor	Hum-TC-Hyg-1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-TC-Xhab-1-Sensor	Hum-TC-Xhab-1-Sensor	Hum-TC-Xhab-1-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
Hum-Temp-B1-Sensor	Hum-Temp-B1-Sensor	Hum-Temp-B1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-Temp-C1-Sensor	Hum-Temp-C1-Sensor	Hum-Temp-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-Temp-D1-Sensor	Hum-Temp-D1-Sensor	Hum-Temp-D1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Hum-Xhab-1-Sensor	Hum-Xhab-1-Sensor	Hum-Xhab-1-Sensor, 24VDC-Power, B1-Power,

		Wireless-Data,
TC-A1-Sensor	TC-A1-Sensor	TC-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A2-Sensor	TC-A2-Sensor	TC-A2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A3-Sensor	TC-A3-Sensor	TC-A3-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A4-Sensor	TC-A4-Sensor	TC-A4-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A5-Sensor	TC-A5-Sensor	TC-A5-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A-L-2-Sensor	TC-A-L-2-Sensor	TC-A-L-2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-A-L-3-Sensor	TC-A-L-3-Sensor	TC-A-L-3-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-B1-Sensor	TC-B1-Sensor	TC-B1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-B2-Sensor	TC-B2-Sensor	TC-B2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-C1-Sensor	TC-C1-Sensor	TC-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-C2-Sensor	TC-C2-Sensor	TC-C2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-C3-Sensor	TC-C3-Sensor	TC-C3-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-C4-Sensor	TC-C4-Sensor	TC-C4-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-D1-Sensor	TC-D1-Sensor	TC-D1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-D2-Sensor	TC-D2-Sensor	TC-D2-Sensor,

		24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-Dust-G5-Sensor	TC-Dust-G5-Sensor	TC-Dust-G5-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-E1-Sensor	TC-E1-Sensor	TC-E1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-E2-Sensor	TC-E2-Sensor	TC-E2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-E3-Sensor	TC-E3-Sensor	TC-E3-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-E4-Sensor	TC-E4-Sensor	TC-E4-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-E5-Sensor	TC-E5-Sensor	TC-E5-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-F1-Sensor	TC-F1-Sensor	TC-F1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-F2-Sensor	TC-F2-Sensor	TC-F2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-G1-Sensor	TC-G1-Sensor	TC-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-G2-Sensor	TC-G2-Sensor	TC-G2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-G3-Sensor	TC-G3-Sensor	TC-G3-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-G4-Sensor	TC-G4-Sensor	TC-G4-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-G5-Sensor	TC-G5-Sensor	TC-G5-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-H1-Sensor	TC-H1-Sensor	TC-H1-Sensor, 24VDC-Power, Wireless-Data,

		B1-28VDC-Power,
TC-H2-Sensor	TC-H2-Sensor	TC-H2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-Hyg-1-Sensor	TC-Hyg-1-Sensor	TC-Hyg-1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-Hyg-2-Sensor	TC-Hyg-2-Sensor	TC-Hyg-2-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
TC-Xhab-1-Sensor	TC-Xhab-1-Sensor	TC-Xhab-1-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-2-Sensor	TC-Xhab-2-Sensor	TC-Xhab-2-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-3-Sensor	TC-Xhab-3-Sensor	TC-Xhab-3-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-4-Sensor	TC-Xhab-4-Sensor	TC-Xhab-4-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-5-Sensor	TC-Xhab-5-Sensor	TC-Xhab-5-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-6-Sensor	TC-Xhab-6-Sensor	TC-Xhab-6-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-7-Sensor	TC-Xhab-7-Sensor	TC-Xhab-7-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-8-Sensor	TC-Xhab-8-Sensor	TC-Xhab-8-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
TC-Xhab-9-Sensor	TC-Xhab-9-Sensor	TC-Xhab-9-Sensor, 24VDC-Power, B1-Power, Wireless-Data,
Veg-Hum-A1-Sensor	Veg-Hum-A1-Sensor	Veg-Hum-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-C1-Sensor	Veg-Hum-C1-Sensor	Veg-Hum-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-E1-Sensor	Veg-Hum-E1-Sensor	Veg-Hum-E1-Sensor,

		24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-G1-Sensor	Veg-Hum-G1-Sensor	Veg-Hum-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-TC-A1-Sensor	Veg-Hum-TC-A1-Sensor	Veg-Hum-TC-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-TC-C1-Sensor	Veg-Hum-TC-C1-Sensor	Veg-Hum-TC-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-TC-E1-Sensor	Veg-Hum-TC-E1-Sensor	Veg-Hum-TC-E1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Hum-TC-G1-Sensor	Veg-Hum-TC-G1-Sensor	Veg-Hum-TC-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Photo-A1-Sensor	Veg-Photo-A1-Sensor	Veg-Photo-A1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Photo-C1-Sensor	Veg-Photo-C1-Sensor	Veg-Photo-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Photo-E1-Sensor	Veg-Photo-E1-Sensor	Veg-Photo-E1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-Photo-G1-Sensor	Veg-Photo-G1-Sensor	Veg-Photo-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-A1-Sensor	Veg-TC-A1-Sensor	Veg-TC-A1-Sensor, 24VDC-Power, B1-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-B1-Sensor	Veg-TC-B1-Sensor	Veg-TC-B1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-C1-Sensor	Veg-TC-C1-Sensor	Veg-TC-C1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-D1-Sensor	Veg-TC-D1-Sensor	Veg-TC-D1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-E1-Sensor	Veg-TC-E1-Sensor	Veg-TC-E1-Sensor, 24VDC-Power,

		Wireless-Data, B1-28VDC-Power,
Veg-TC-F1-Sensor	Veg-TC-F1-Sensor	Veg-TC-F1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-G1-Sensor	Veg-TC-G1-Sensor	Veg-TC-G1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,
Veg-TC-H1-Sensor	Veg-TC-H1-Sensor	Veg-TC-H1-Sensor, 24VDC-Power, Wireless-Data, B1-28VDC-Power,

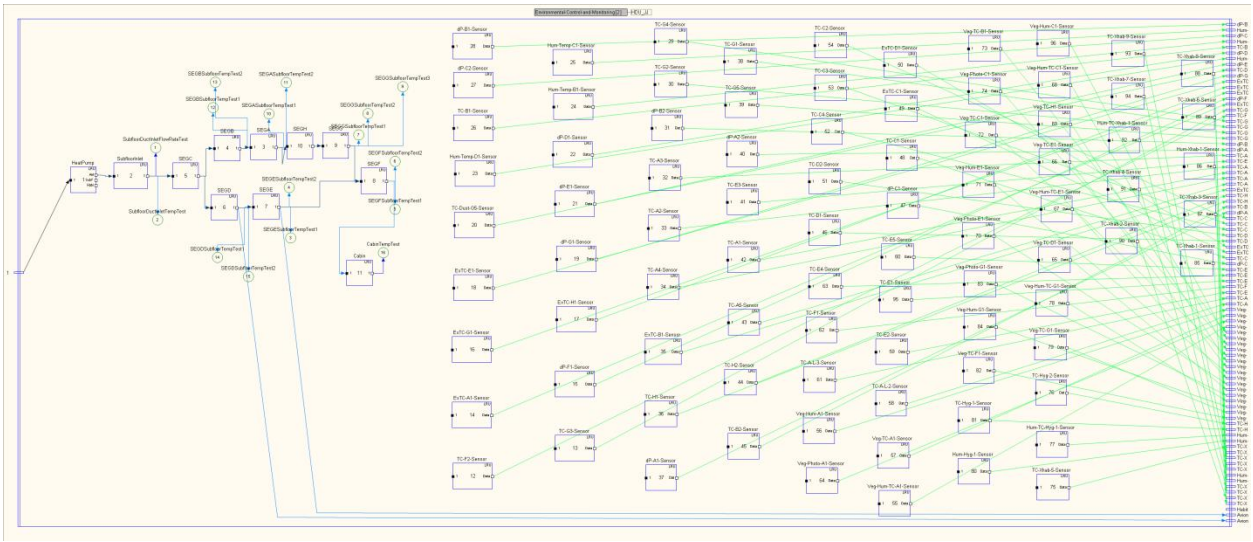

Figure 16: Environmental Control and Monitoring

TEAMS Module	Failure Mode	Failure Signals
Cabin	CoolingFailure	Heat
dP-A1-Sensor	dP-A1-Sensor-Failure	dP-A1-Sensor
dP-A2-Sensor	dP-A2-Sensor-Failure	dP-A2-Sensor
dP-B1-Sensor	dP-B1-Sensor-Failure	dP-B1-Sensor
dP-B2-Sensor	dP-B2-Sensor-Failure	dP-B2-Sensor
dP-C1-Sensor	dP-C1-Sensor-Failure	dP-C1-Sensor
dP-C2-Sensor	dP-C2-Sensor-Failure	dP-C2-Sensor
dP-D1-Sensor	dP-D1-Sensor-Failure	dP-D1-Sensor
dP-E1-Sensor	dP-E1-Sensor-Failure	dP-E1-Sensor
dP-F1-Sensor	dP-F1-Sensor-Failure	dP-F1-Sensor
dP-G1-Sensor	dP-G1-Sensor-Failure	dP-G1-Sensor
ExTC-A1-Sensor	ExTC-A1-Sensor-Failure	ExTC-A1-Sensor
ExTC-B1-Sensor	ExTC-B1-Sensor-Failure	ExTC-B1-Sensor
ExTC-C1-Sensor	ExTC-C1-Sensor-Failure	ExTC-C1-Sensor

ExTC-D1-Sensor	ExTC-D1-Sensor-Failure	ExTC-D1-Sensor
ExTC-E1-Sensor	ExTC-E1-Sensor-Failure	ExTC-E1-Sensor
ExTC-G1-Sensor	ExTC-G1-Sensor-Failure	ExTC-G1-Sensor
ExTC-H1-Sensor	ExTC-H1-Sensor-Failure	ExTC-H1-Sensor
HeatPump	HeatPumpFailure	Heat HeatPumpFailure Power
Hum-Hyg-1-Sensor	Hum-Hyg-1-Sensor-Failure	Hum-Hyg-1-Sensor
Hum-TC-Hyg-1-Sensor	Hum-TC-Hyg-1-Sensor-Failure	Hum-TC-Hyg-1-Sensor
Hum-TC-Xhab-1-Sensor	Hum-TC-Xhab-1-Sensor-Failure	Hum-TC-Xhab-1-Sensor
Hum-Temp-B1-Sensor	Hum-Temp-B1-Sensor-Failure	Hum-Temp-B1-Sensor
Hum-Temp-C1-Sensor	Hum-Temp-C1-Sensor-Failure	Hum-Temp-C1-Sensor
Hum-Temp-D1-Sensor	Hum-Temp-D1-Sensor-Failure	Hum-Temp-D1-Sensor
Hum-Xhab-1-Sensor	Hum-Xhab-1-Sensor-Failure	Hum-Xhab-1-Sensor
SEGA	Obstruction_SEGA	Heat
SEGB	Obstruction_SEGB	Heat
SEGC	Obstruction_SEGC	Heat
SEGD	Obstruction_SEGD	Data Heat Obstruction_SEGD Power
SEGE	Obstruction_SEGE	Data Heat Obstruction_SEGE Power
SEGF	Obstruction_SEGF	Heat
SEGG	Obstruction_SEGG	Heat
SEGH	Obstruction_SEGH	Heat
SubfloorInlet	InletObstruction	Heat
TC-A1-Sensor	TC-A1-Sensor-Failure	TC-A1-Sensor
TC-A2-Sensor	TC-a2-Sensor-Failure	TC-A2-Sensor
TC-A3-Sensor	TC-A3-Sensor-Failure	TC-A3-Sensor
TC-A4-Sensor	TC-A4-Sensor-Failure	TC-A4-Sensor
TC-A5-Sensor	TC-A5-Sensor-Failure	TC-A5-Sensor
TC-A-L-2-Sensor	TC-A-L-2-Sensor-Failure	TC-A-L-2-Sensor
TC-A-L-3-Sensor	TC-A-L-3-Sensor-Failure	TC-A-L-3-Sensor
TC-B1-Sensor	TC-B1-Sensor-Failure	TC-B1-Sensor
TC-B2-Sensor	TC-B2-Sensor-Failure	TC-B2-Sensor
TC-C1-Sensor	TC-C1-Sensor-Failure	TC-C1-Sensor
TC-C2-Sensor	TC-C2-Sensor-Failure	TC-C2-Sensor
TC-C3-Sensor	TC-C3-Sensor-Failure	TC-C3-Sensor
TC-C4-Sensor	TC-C4-Sensor-Failure	TC-C4-Sensor
TC-D1-Sensor	TC-D1-Sensor-Failure	TC-D1-Sensor
TC-D2-Sensor	TC-D2-Sensor-Failure	TC-D2-Sensor
TC-Dust-G5-Sensor	TC-Dust-G5-Sensor-Failure	TC-Dust-G5-Sensor
TC-E1-Sensor	TC-E1-Sensor-Failure	TC-E1-Sensor
TC-E2-Sensor	TC-E2-Sensor-Failure	TC-E2-Sensor
TC-E3-Sensor	TC-E3-Sensor-Failure	TC-E3-Sensor
TC-E4-Sensor	TC-E4-Sensor-Failure	TC-E4-Sensor
TC-E5-Sensor	TC-E5-Sensor-Failure	TC-E5-Sensor
TC-F1-Sensor	TC-F1-Sensor-Failure	TC-F1-Sensor
TC-F2-Sensor	TC-F2-Sensor-Failure	TC-F2-Sensor
TC-G1-Sensor	TC-G1-Sensor-Failure	TC-G1-Sensor
TC-G2-Sensor	TC-G2-Sensor-Failure	TC-G2-Sensor
TC-G3-Sensor	TC-G3-Sensor-Failure	TC-G3-Sensor
TC-G4-Sensor	TC-G4-Sensor-Failure	TC-G4-Sensor

TC-G5-Sensor	TC-G5-Sensor-Failure	TC-G5-Sensor
TC-H1-Sensor	TC-H1-Sensor-Failure	TC-H1-Sensor
TC-H2-Sensor	TC-H2-Sensor-Failure	TC-H2-Sensor
TC-Hyg-1-Sensor	TC-Hyg-1-Sensor-Failure	TC-Hyg-1-Sensor
TC-Hyg-2-Sensor	TC-Hyg-2-Sensor-Failure	TC-Hyg-2-Sensor
TC-Xhab-1-Sensor	TC-Xhab-1-Sensor-Failure	TC-Xhab-1-Sensor
TC-Xhab-2-Sensor	TC-Xhab-2-Sensor-Failure	TC-Xhab-2-Sensor
TC-Xhab-3-Sensor	TC-Xhab-3-Sensor-Failure	TC-Xhab-3-Sensor
TC-Xhab-4-Sensor	TC-Xhab-4-Sensor-Failure	TC-Xhab-4-Sensor
TC-Xhab-5-Sensor	TC-Xhab-5-Sensor-Failure	TC-Xhab-5-Sensor
TC-Xhab-6-Sensor	TC-Xhab-6-Sensor-Failure	TC-Xhab-6-Sensor
TC-Xhab-7-Sensor	TC-Xhab-7-Sensor-Failure	TC-Xhab-7-Sensor
TC-Xhab-8-Sensor	TC-Xhab-8-Sensor-Failure	TC-Xhab-8-Sensor
TC-Xhab-9-Sensor	TC-Xhab-9-Sensor-Failure	TC-Xhab-9-Sensor
Veg-Hum-A1-Sensor	Veg-Hum-A1-Sensor-Failure	Veg-Hum-A1-Sensor
Veg-Hum-C1-Sensor	Veg-Hum-C1-Sensor-Failure	Veg-Hum-C1-Sensor
Veg-Hum-E1-Sensor	Veg-Hum-E1-Sensor-Failure	Veg-Hum-E1-Sensor
Veg-Hum-G1-Sensor	Veg-Hum-G1-Sensor-Failure	Veg-Hum-G1-Sensor
Veg-Hum-TC-A1-Sensor	Veg-Hum-TC-A1-Sensor-Failure	Veg-Hum-TC-A1-Sensor
Veg-Hum-TC-C1-Sensor	Veg-Hum-TC-C1-Sensor-Failure	Veg-Hum-TC-C1-Sensor
Veg-Hum-TC-E1-Sensor	Veg-Hum-TC-E1-Sensor-Failure	Veg-Hum-TC-E1-Sensor
Veg-Hum-TC-G1-Sensor	Veg-Hum-TC-G1-Sensor-Failure	Veg-Hum-TC-G1-Sensor
Veg-Photo-A1-Sensor	Veg-Photo-A1-Sensor-Failure	Veg-Photo-A1-Sensor
Veg-Photo-C1-Sensor	Veg-Photo-C1-Sensor-Failure	Veg-Photo-C1-Sensor
Veg-Photo-E1-Sensor	Veg-Photo-E1-Sensor-Failure	Veg-Photo-E1-Sensor
Veg-Photo-G1-Sensor	Veg-Photo-G1-Sensor-Failure	Veg-Photo-G1-Sensor
Veg-TC-A1-Sensor	Veg-TC-A1-Sensor-Failure	Veg-TC-A1-Sensor
Veg-TC-B1-Sensor	Veg-TC-B1-Sensor-Failure	Veg-TC-B1-Sensor
Veg-TC-C1-Sensor	Veg-TC-C1-Sensor-Failure	Veg-TC-C1-Sensor
Veg-TC-D1-Sensor	Veg-TC-D1-Sensor-Failure	Veg-TC-D1-Sensor
Veg-TC-E1-Sensor	Veg-TC-E1-Sensor-Failure	Veg-TC-E1-Sensor
Veg-TC-F1-Sensor	Veg-TC-F1-Sensor-Failure	Veg-TC-F1-Sensor
Veg-TC-G1-Sensor	Veg-TC-G1-Sensor-Failure	Veg-TC-G1-Sensor
Veg-TC-H1-Sensor	Veg-TC-H1-Sensor-Failure	Veg-TC-H1-Sensor

TEAMS Testpoint	TEAMS Test	Failure Signals
CabinTempTest	CabinTempTest	Heat
SEGASubfloorTempTest1	SEGASubfloorTempTest1	Heat
SEGASubfloorTempTest2	SEGASubfloorTempTest2	Heat
SEGBSubfloorTempTest1	SEGBSubfloorTempTest1	Heat
SEGBSubfloorTempTest2	SEGBSubfloorTempTest2	Heat
SEGDSubfloorTempTest1	SEGDSubfloorTempTest1	Heat
SEGDSubfloorTempTest2	SEGDSubfloorTempTest2	Heat
SEGESubfloorTempTest1	SEGESubfloorTempTest1	Heat
SEGESubfloorTempTest2	SEGESubfloorTempTest2	Heat
SEGFSubfloorTempTest1	SEGFSubfloorTempTest1	Heat
SEGFSubfloorTempTest2	SEGFSubfloorTempTest2	Heat
SEGGSubfloorTempTest1	SEGGSubfloorTempTest1	Heat
SEGGSubfloorTempTest2	SEGGSubfloorTempTest2	Heat
SEGGSubfloorTempTest3	SEGGSubfloorTempTest3	Heat
SubfloorDuctInletFlowRateTest	SubfloorDuctInletFlowRateTest	Heat
SubfloorDuctInletTempTest	SubfloorDuctInletTempTest	Heat

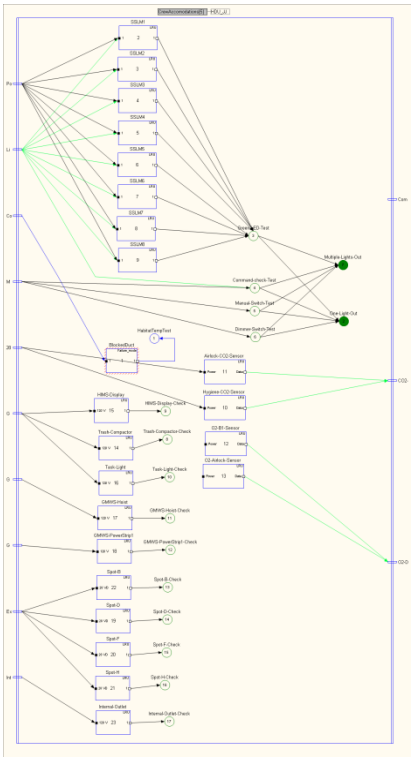

Figure 17: Crew Accommodations

TEAMS Module	Failure Mode	Failure Signals
Airlock-CO2-Sensor	Airlock-CO2-Sensor-failure	Airlock-CO2
CrewAccomodations	BlockedDuct	Heat
GMWS-Hoist	GMWS-Hoist-failure	GMWS-Hoist-Out
GMWS-PowerStrip1	GMWS-PowerStrip1-failure	GMWS-PowerStrip1-Out
HIMS-Display	HIMS-Display-failure	HIMS-Display-Out
Hygiene-CO2-Sensor	Hygiene-CO2-Sensor-failure	Hygiene-CO2
Internal-Outlet	Internal-Outlet-failure	Internal-Outlet-Out
O2-Airlock-Sensor	O2-Airlock-Sensor-failure	O2-Airlock
O2-B1-Sensor	O2-B1-Sensor-failure	O2-B1
Spot-B	Spot-B-failure	Spot-B-Out
Spot-D	Spot-D-failure	Spot-D-Out
Spot-F	Spot-F-failure	Spot-F-Out
Spot-H	Spot-H-failure	Spot-H-Out
SSLM1	SSLM1-failure	no-light no-light-SSLM1
SSLM2	SSLM2-failure	no-light no-light-SSLM2
SSLM3	SSLM3-failure	no-light no-light-SSLM3
SSLM4	SSLM4-failure	no-light no-light-SSLM4
SSLM5	SSLM5-failure	no-light

		no-light-SSLM5
SSLM6	SSLM6-failure	no-light no-light-SSLM6
SSLM7	SSLM7-failure	no-light no-light-SSLM7
SSLM8	SSLM8-failure	no-light no-light-SSLM8
Task-Light	Task-Light-failure	Task-Light-Out
Trash-Compactor	Trash-Compactor-failure	Trash-Compactor-Out

TEAMS Testpoint	**TEAMS Test**	**Failure Signals**
Command-check-Test	Lights-commanded-on-ok	command-is-off,
Command-check-Test	Light-Command-Response-Test	light-command,
Dimmer-Switch-Test	Dimmer-Switch-is-bright-ok	unintended-dimming-command,
GMWS-Hoist-Check	GMWS-Hoist-Check	GMWS-Hoist-Out, Power,
GMWS-PowerStrip1-Check	GMWS-PowerStrip1-Check	GMWS-PowerStrip1-Out, Power,
Green-LED-Test	Power-to-SSLM1-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM2-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM3-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM4-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM5-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM6-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM7-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
Green-LED-Test	Power-to-SSLM8-ok	120DC-Power, PDU2Failure, JunctionBoxFailure, SourceFailure,
HabitatTempTest	HabitatTempTest	Heat,
HIMS-Display-Check	HIMS-Display-Check	HIMS-Display-Out, Power,
Internal-Outlet-Check	Internal-Outlet-Check	Internal-Outlet-Out, Power,

Manual-Switch-Test	Manual-Switch-is-on-ok	unintended-manual-off-command,
Multiple-Lights-Out	Multiple-Lights-Out	unintended-dimming-command, unintended-manual-off-command, no-light, command-is-off, CDHSwitchFailure, PDUFailure, Obstruction_SEGD, Obstruction_SEGE, HeatPumpFailure, SourceFailure, JunctionBoxFailure, PDU2Failure, ACDCFailure, Unintended-External-Power-Switch-Deactivation, PDU2-Bank1-Failure, PDU2-Bank2-Failure,
One-Light-Out	One-Light-Off	unintended-dimming-command, unintended-manual-off-command, command-is-off, no-light-SSLM1, no-light-SSLM2, no-light-SSLM4, no-light-SSLM3, no-light-SSLM5, no-light-SSLM6, no-light-SSLM7, no-light-SSLM8,
Spot-B-Check	Spot-B-Check	Spot-B-Out, B1-28VDC-Power,
Spot-D-Check	Spot-D-Check	Spot-D-Out, B1-28VDC-Power,
Spot-F-Check	Spot-F-Check	Spot-F-Out, B1-28VDC-Power,
Spot-H-Check	Spot-H-Check	Spot-H-Out, B1-28VDC-Power,
Task-Light-Check	Task-Light-Check	Task-Light-Out, Power,
Trash-Compactor-Check	Trash-Compactor-Check	Trash-Compactor-Out, Power,

Failure Scenarios

The ACAWS team identified four demonstration scenarios consisting of failure injection, failure diagnosis and failure recovery. The D-matrices for Scenario 2 (24 Volt Direct Current (VDC) Power Supply Failure) are shown below. The first D-matrix shows the diagnosis without manual tests (check 24 VDC Power Supply LED Light, while the second D-matrix shows the diagnosis with manual tests (see Figure 18). Without manual tests, the diagnosis cannot be narrowed to the 24 VDC power supply and is one of the suspects (yellow rows); with manual tests, however, the diagnosis identifies the fault (see top red row, Figure 19).

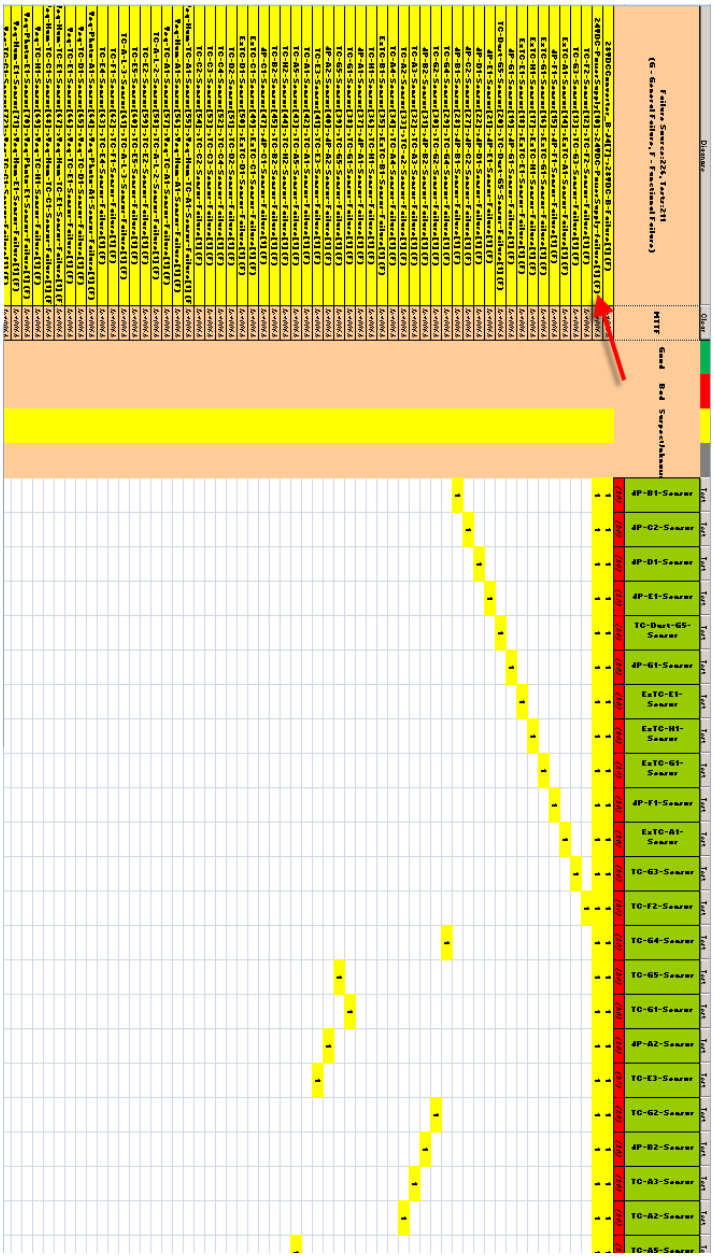

Figure 18: In scenario 2, 24 VDC Power Supply cannot be isolated as the fault without manual observation (test result). Red arrow points to that component in the suspect list.

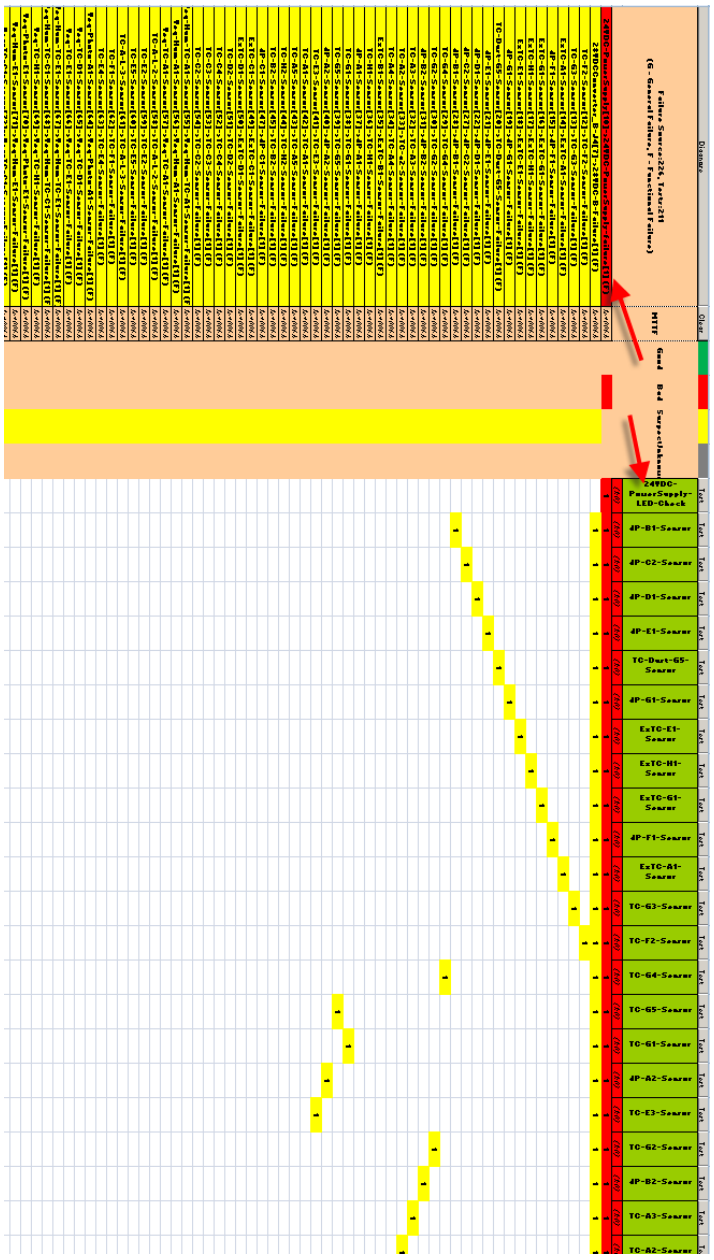

Figure 19: In Scenario 2, failure is correctly isolated to 24 VDC Power Supply using manual observation procedure “Check 24 VDC LED light” (see arrows in diagram).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
19-03-2013	Technical Memorandum	10/2010 - 9/2011

4. TITLE AND SUBTITLE

Advanced Caution and Warning System, Final Report — 2011

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lilly Spirkovska, Gordon Aaseng, David Iverson, Robert S. McCann, Peter Robinson, Gary Dittemore, Sotirios Liolios, Vijay Baskaran, Jeremy Johnson, Charles Lee, John Ossenfort, Mike Dalal, Chuck Fry, and Larry Garner

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

402600.04.03.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Ames Research Center
Moffet Field, California 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITOR'S ACRONYM(S)

NASA

11. SPONSORING/MONITORING REPORT NUMBER

Nasa TM-2013-216510

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified-Unlimited
Subject Category 16
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The work described in this report is a continuation of the ACAWS work funded in fiscal year (FY) 2010 under the Exploration Technology Development Program (ETDP), Integrated Systems Health Management (ISHM) project. In FY 2010, we developed requirements for an ACAWS system and vetted the requirements with potential users via a concept demonstration system. In FY 2011, we developed a working prototype of aspects of that concept, with placeholders for technologies to be fully developed in future phases of the project. The objective is to develop general capability to assist operators with system health monitoring and failure diagnosis. Moreover, ACAWS was integrated with the Discrete Controls (DC) task of the Autonomous Systems and Avionics (ASA) project. The primary objective of DC is to demonstrate an electronic and interactive procedure display environment and multiple levels of automation (automatic execution by computer, execution by computer if the operator consents, and manual execution by the operator).

15. SUBJECT TERMS

warning systems, malfunctions, impact

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19b. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
U	U	U	UU	89	19b. TELEPHONE NUMBER *(Include area code)* (443) 757-5802