NASA/TM-2013-216509

Advanced Caution and Warning System, Final Report — 2010

Lilly Spirkovska
Ames Research Center
Moffett Field, California

Peter Robinson
Ames Research Center
Moffett Field, California

Sotirios Liolios
Johnson Space Center
Houston, Texas

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

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

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-2013-216509

Advanced Caution and Warning System, Final Report — 2010

Lilly Spirkovska
Ames Research Center
Moffett Field, California

Peter Robinson
Ames Research Center
Moffett Field, California

Sotirios Liolios
Johnson Space Center
Houston, Texas

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

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

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 Mission Operations Project
With the Exploration Technology Development Program,
Integrated Systems Health Management Project –

Advanced Caution and Warning System (ACAWS)

Lilly Spirkovska and Peter Robinson
NASA Ames Research Center

Sotirios Liolios
NASA Johnson Space Center

Charles Lee and John Ossenfort
Stinger Ghaffarian Technologies Inc.
at NASA Ames Research Center

Table of Contents

Table of Contents 2
List of Figures 3
List of Tables 5
Preamble 6
1. Introduction 6
2. Background 7
3. Spaceflight Operations 13
4. Development Approach 16
5. Functional Requirements 17
5.1 General 18
5.2 System Monitoring 20
5.3 Failure 23
5.4 Impact 24
5.5 Recovery and Workaround 25
6. Design 25
6.1 High-Level Architecture 26
6.2 Functional Requirements Distribution 29
6.3 Demonstration System Approach 38
6.4 User Interface Design Concept 41
7. Implementation 41
7.1 Investigative Domain 41
7.2 ACAWS Implementation Requirements 41
7.3 ACAWS Implementation Details 42
7.4 Demonstration Scenario 54
8. Lessons Learned 63
8.1 Schematic Diagrams 63
8.2 Diagnostic Models 64
8.3 Data System 64
8.4 User Interface 65
Appendix A: ACAWS Team 67
Appendix B: Acronym List 68
Appendix C: ACAWS Framework Design 70
Appendix D: Notes 88

List of Figures

Figure 1: Example of MSK-View display. 8
Figure 2: Example of RT-Plot display. 9
Figure 3: Example of ELOG (Event Logger) display. 9
Figure 4: Example of schematic from System Handbook. 10
Figure 5: Example of CRANS display. 10
Figure 6: Mission Control Technologies (MCT) concept. 11
Figure 7: Two different views of mission operations state: now and with ACAWS integrated with MCT. 26
Figure 8: ACAWS high-level architecture. 26
Figure 9: ACAWS subsystems and categories of capabilities. 28
Figure 10: Grouping and prioritization of ACAWS demo capabilities. The eight groups are shown as pieces of the pie (i.e., oval), with the labels in bold on the outside of the pie. The highest priority items are in the inner yellow core. Priority of the outer rings decreases as the distances from the center increases. The priority of mocked-up items ("smoke" for "smoke and mirrors" implementation approach) and storyboarded ("story") items is color-coded and distinguished by the type of line around each rectangle. 38
Figure 11: Grouping and prioritization of ACAWS demo capabilities, different view but same information as Figure 10. The eight groups are shown in four columns and two groups with the delineation mark between the upper and lower group shown going between the two yellow rows. The upper groups have the priority 1 items at the bottom whereas the lower groups have them at the top. In this way, we can show the highest priority items together and priority decreases in both (up/down) directions from the middle. 39
Figure 12: Implementation Architecture 44
Figure 13: TVC Overview Block Diagram 45
Figure 14: TEAMS Model Hierarchy 46
Figure 15: TEAMS Model, top-level view 47
Figure 16: TEAMS Designer Model is constructed from Avionics and TVC models. The model is hierarchical with distributed failure modes and test points. The full model is compiled into a D-matrix (one per system mode). 48
Figure 17: TEAMS Designer Model Converted to D-matrix. 49
Figure 18: TEAMS D-Matrix for Ascent Phase. 49
Figure 19: TEAMS RT Wrapper Specification in Test Description Format (TDF). 50
Figure 20: ACAWS Default Layout. 51
Figure 21: Post-diagnosis ACAWS Display 53
Figure 22: Step 1 – 4: Prelaunch Relevant MSIDs. Relevant portions of Actuator Position (prelaunch), relevant portions of SRB CH ORIDE – prelaunch override. 58
Figure 23: Steps 1 – 4: TEAMS Diagnosis – No Faults. 58
Figure 24: Step 5 – 7: First Failure MSIDs. When FA-1 MDM BCE-Bypass Flag goes high, then the control path to the actuator is cut. Actuator Driver current gets stuck at constant and servo valve delta-pressure sensor starts to increase as force-fight with remaining three channels ensues. 59

Figure 25: Steps 5 -7 GMT: 221:22:16:26.0; MET: 13.6 TEAMS Diagnosis First Failure- FA-1 MDM Failed. 59
Figure 26: Crew recovery from first failure: take Channel 1 Offline. 60
Figure 27: Steps 9 – 11: Second Failure MSIDs. 60
Figure 28: Steps 9 – 11. 61
Figure 29: Steps 9 -11 221:22:17:14.0; MET Seconds: 59.9204 TEAMS Diagnosis First Faillure + Second Failure: FA-1 MDM Failed, ATVC_Driver_Ch2_Failed 62
Figure 30: Crew recovery from second failure: take Channels 3 and 4 to override. 62
Figure 31: ACAWS application window layout. 70
Figure 32: Window layout preferences details. 70
Figure 33: Window layout details. 71
Figure 34: Window header details. 71
Figure 35: Panel details. 72
Figure 36: All panels shown. 72
Figure 37: Window configuration for training (example). 73
Figure 38: Window configuration for troubleshooting (example). 73
Figure 39: Window configuration for monitoring (example). 74
Figure 40: Window configuration for analysis (example). 74
Figure 41: Block diagram design. 75
Figure 42: Block diagram details. 75
Figure 43: Block diagram menu details. 76
Figure 44: Block diagram values matrix details. 76
Figure 45: Block diagram folder details. 77
Figure 46: System health annunciators panel. 77
Figure 47: System health annunciators panel details. 78
Figure 48: System health annunciators panel scroll window details. 78
Figure 49: ELOG messages panel. 79
Figure 50: ELOG messages panel details. 79
Figure 51: ELOG messages panel details, alternate color-coding. 80
Figure 52: Procedures panel. 81
Figure 53: Procedures panel details. 81
Figure 54: Flight rules panel. 82
Figure 55: Flight rules panel details. 82
Figure 56: Diagnosis panel. 83
Figure 57: Diagnosis panel details. 83
Figure 58: Systems impact panel. 84
Figure 59: Systems impact panel details. 84
Figure 60: Systems impact panel, more details. 85
Figure 61: Mission impact panel. 86
Figure 62: Mission impact panel details. 86
Figure 63: MSK tabular display panel. 87
Figure 64: Trend plot display panel. 87

List of Tables

Table 1: Spaceflight operations roles and associated tasks. 14
Table 2: ACAWS development phases – capabilities and schedule. 16
Table 3: Demonstration column color-coding and character key. 29
Table 4: Generic requirements. 29
Table 5: Model Manager -- Merge Models capability. 30
Table 6: Model Manager -- Modify Models capability. 31
Table 7: Model Manager -- Augment Models capability. 31
Table 8: Model Manager -- Save/Load Models capability. 31
Table 9: Model Manager -- Search Models Database capability. 32
Table 10: Root Cause Manager -- Health Status capability. 32
Table 11: Root Cause Manager -- Root Cause capability. 32
Table 12: Root Cause Manager -- Group/Order capability. 32
Table 13: Root Cause Manager -- Rationale capability. 32
Table 14: Root Cause Manager -- Troubleshooting capability. 32
Table 15: System Impact Manager – System Impact Advisor capability. 33
Table 16: System Impact Manager -- Flight Rules Liaison capability. 33
Table 17: Mission Impact Manager. 33
Table 18: Recovery Procedure Manager -- Procedure Advisor capability. 34
Table 19: Recovery Procedure Manager -- Flight Note Liaison capability. 34
Table 20: Problem Reporting Liaison. 34
Table 21: Data Manager. 34
Table 22: User Interface -- View ACAWS Model capability. 35
Table 23: User Interface -- View Health Information capability. 35
Table 24: User Interface -- View Telemetry capability. 36
Table 25: User Interface -- Overlay Information capability. 36
Table 26: User Interface -- Display ACAWS Status/Configuration capability. 37
Table 27: User Interface -- Navigation capability. 37
Table 28: User Interface -- Save/Load Preferences/Viewing Configurations capability. 38
Table 29: Unclassified requirements. 38
Table 30: Schematic Diagrams. 41
Table 31: Features of TEAMS that are utilized for ACAWS prototype development. 42
Table 32: Requirements for extensions to TEAMS to enable ACAWS. 42
Table 33: Requirements for UI and data system. 42
Table 34: Demonstration scenario timeline of events. 54
Table 35: Demonstration scenario timeline of events with TEAMS and MSID strip chart annotations. 55

For the Mission Operations Project
With the Exploration Technology Development Program, Integrated Systems Health Management Project – *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.

As described below, the team is employing a phased development approach. We are not starting with a complete list of user needs. Rather, each phase of development has its own requirements definition portion. For phase A, we are concentrating on obtaining a solid set of requirements. We also are including ideas for the subsequent phases. These ideas are more nebulous at this stage and will be clarified further during the next phases but are included in the initial phase to better define the direction we are heading and decrease the amount of possible rework due to unanticipated complexity.

Version	Date	Description
1	April 1, 2010	Understanding of the project as of the end of the requirements development portion for the phase A prototype.
2	September 30, 2010	Final report for ETDP.

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 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 supports 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 current focus of ACAWS is on the needs of the flight controllers. The onboard crew in low-Earth orbit has some of those same needs. Moreover, for future deep-space missions, the crew will need to accomplish many tasks autonomously due to communication time delays. Although we are focusing on flight controller needs, ACAWS technologies can be reused for on-board application, perhaps with a different level of detail and different display formats or interaction methods. We expect that providing similar tools 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* is used to refer to either flight controllers or crew.

2. Background

The Mission Operations Directorate (MOD) is involved in three main activities: planning for a mission, training for a mission, and executing/flying a mission. These *plan – train – fly* activities are performed at various levels and occur throughout the entire mission profile. These activities take place prior to launch, during the launch and mission execution phase as well as post-flight.

- The planning process includes, for example, development of: mission timelines, consumables analyses, crew procedures, training products, simulators, and software loads. Planning occurs at three levels: strategic (greater than one year prior to mission start), tactical (from one year to one month prior to mission start), and execution (one month prior to mission start through execution).
- MOD conducts training and certification necessary for space flight crews, flight controllers, analysts, instructors, and other identified personnel to successfully operate a vehicle. Training covers flight specific team training, as well as the skills required for nominal and contingency operations of spacecraft, cargo/payload/DTO[1] hardware and ground systems, execution of the planned timeline, and the identification of and response to operational issues and malfunctions. Once an operator is certified, training is also performed for the purpose of maintaining proficiency.

[1] DTO = Development Test Objective

- Flight operations are defined as all activities associated with the operations of the flight vehicle, crew, and operations teams that support the flight vehicle from pre-launch until landing. Flight operations are supported as needed during the execution of a mission. Examples of flight tasks include command and control of the vehicle; in-flight mission performance analysis and mission planning and re-planning; and vehicle and mission anomaly tracking, analysis, troubleshooting, and resolution.

Operators utilize a number of independent tools to support the *plan – train – fly* tasks. Of particular importance to the ACAWS team are the products and tools utilized in the current approach to caution and warning (C&W). Operators combine the information gathered from the following C&W tools with their analysis and reasoning developed through extensive training to assess spacecraft health:

- MSK-View[2], shown in Figure 1, provides real-time spacecraft telemetry values. The tool is used to display Information Sharing Protocol (ISP) data in a tabular format as defined from a display file. Operators can also associate alarm triggers, known as *limits,* with each parameter. By setting these limits to values lower than those onboard the spacecraft, the vehicle operators are notified of a possible fault before the crew is alerted.

Figure 1: Example of MSK-View display. (Blurred to hide details.)

- RT-Plot, shown in Figure 2, presents the operator with graphed trending data and allows an operator to be proactive rather than just reactive to the occurrence of faults. By observing plots and trends of telemetry values, operators can, for example, identify slow degradation in a system or a slow leak even before alarms are triggered. If a leak is suspected, RT-Plot can also be used to determine the leak rate.

[2] MSK = Manual Select Keyboard; from Apollo program where different displays could be manually selected via keyboard. Now refers to displays in general.

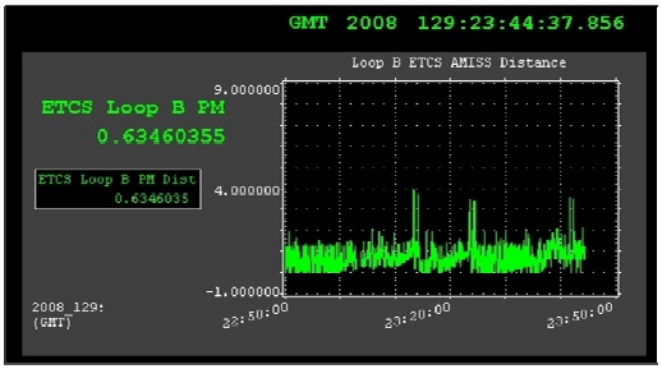


Figure 2: Example of RT-Plot display.

- ELOG (Event Logger) provides a means to automatically log telemetry change events in real time. In ELOG, an event is defined as the comparison (greater than, equal to, etc.) between an ISP parameter and a constant (number, text string, or parent word). When the comparison is true, the event (tagged by GMT, MET, and other user-defined parameters) is logged to a file and displayed, as shown in Figure 3.

Figure 3: Example of ELOG (Event Logger) display. (Blurred to hide details.)

- System Handbook drawings, shown in Figure 4, are generated and maintained by the operations community to provide an operations-unique depiction of system configuration, connectivity, and sensor/effector locations and characteristics. These drawings are used as both training media and as on-console reference material supporting real-time troubleshooting. The current System Handbook development process requires a significant amount of MOD effort to interpret data delivered by the spacecraft vendor, as well as employment of a Computer Aided Design (CAD) staff to develop and maintain the final customized drawings. The source materials for the System Handbook are element-approved drawings.

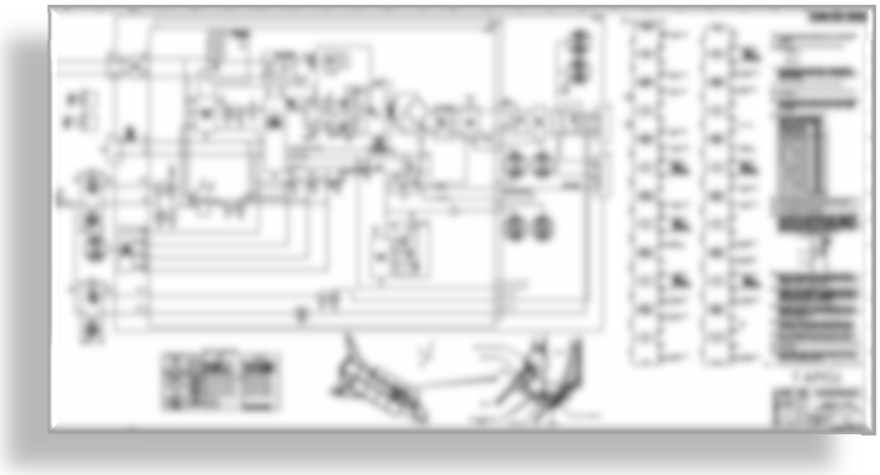


Figure 4: Example of schematic from System Handbook.

- Configurable Real-time Analysis System (CRANS) tool, shown in Figure 5, uses expert system technology, in the form of rule-based logic files, to assess vehicle telemetry and identify failure conditions. These logic files are manually generated by operators based on extensive study of vehicle system architecture, failure mode documentation and simulation, and flight experience. The same CRANS system is also used in an off-line "what if" mode to investigate the impacts of follow-on malfunctions and the potential failure scenarios. ISS experience indicates that maintenance of these data sets for an ever-changing vehicle configuration is difficult and costly because the knowledge required for CRANS cannot be automatically translated from reference material such as RECON[3] products, system handbook products or others.

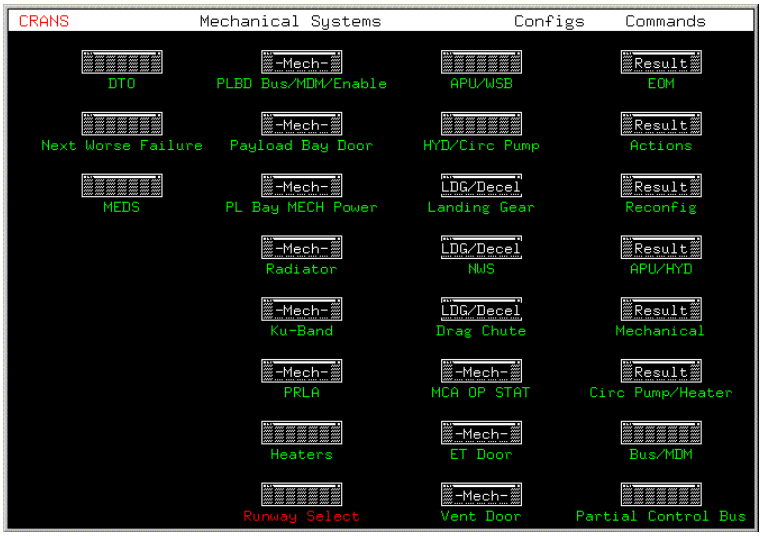


Figure 5: Example of CRANS display.

[3] RECON is the "Reconfiguration" group in MOD. RECON is responsible for maintaining the data products that change with every flight. MOD, training and Ground all use the RECON product to stay consistent.

- Anomaly Monitoring Inductive Software System (AMISS) is a health monitoring software tool that learns system behavior from data. It is based on the NASA/Ames-developed Inductive Monitoring System (IMS) tool which utilizes techniques from the fields of model-based reasoning, machine learning, and data mining to build system monitoring knowledge bases from archived or simulated sensor data. AMISS/IMS automatically analyzes nominal system data to form general classes of expected combinations of system sensor values. These classes are used to build a monitoring knowledge base. When monitoring a system, AMISS/IMS simply checks to see how well the incoming sensor data fits into the classes derived from the training data.[4] AMISS is a fairly new tool being used for ISS monitoring. It can be considered a computation (albeit, a sophisticated one) and, thus, can be treated similarly to other computed parameters (i.e., plotted via RT-Plot, monitored on a tabular display via MSK, etc.).

Each product and tool is managed separately: drawings are maintained as static paper products and CRANS, MSK and similar tools require separate data files.

The Mission Control Technologies (MCT) program currently under development aims to establish a standardized platform for the development and reuse of user tool software. Utility programs, data sets, databases, access to telemetry, and other necessary components will be integrated into one standard toolset that will enable operations personnel to build customized user interfaces and use them to work collaboratively in real-time while increasing situational awareness and decreasing the time it takes to resolve a malfunction. Much of the infrastructure development and sharing of data between tools can be provided to tools that are integrated into the MCT toolset. The concept for MCT is shown in Figure 6.

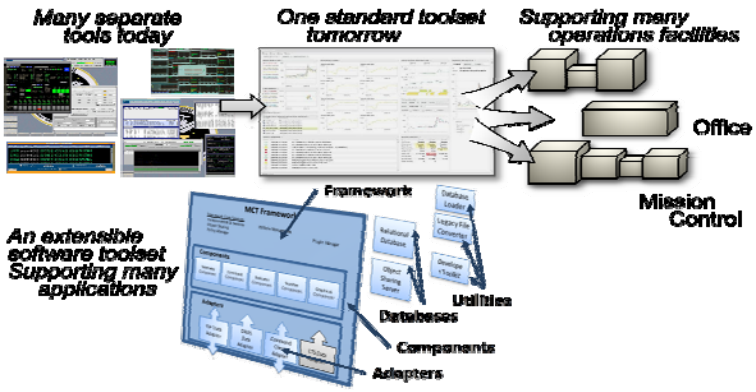


Figure 6: Mission Control Technologies (MCT) concept.

Currently, the functionality of MSK, RT-Plot, and ELOG tools will be integrated within MCT. However, although system schematics can be displayed and CRANS can be executed from within MCT, the two data sets will not be merged into one integrated tool. That is,

[4] IMS was renamed to AMISS for MOD applications to avoid conflict with the existing Inventory Management System (IMS).

schematics can be displayed as a static electronic (rather than paper) product and the CRANS logic files would still need to be developed manually. ACAWS aims to integrate these two products and facilitate the generation and assessment of spacecraft system failure mode and effect information.

Once a system problem has been identified, several tools exist or are under development to assist vehicle operators with fault management, including the following:

- The Problem Reporting and Corrective Action (PRACA) system is a repository for all reportable problems and ultimately the nonconformance data for the purpose of assuring the adequate resolution and oversight of the problems. This is an existing system that was updated for the Constellation Program with the CxPRACA tool.
- The Constellation Procedures Application Software Suite (CxPASS) is a suite of software tools under development to support ground operations for Crew Exploration Vehicle (CEV; aka Orion) electronic procedures. The ground support toolset also includes a procedure authoring tool, two ground based procedure viewers, a procedure library administration tool and a utility for translation and verification of procedures into the CEV onboard procedure and automation script formats.
- Constraint and Flight Rule Management (ConFRM) is a tool to capture operations-related constraint products (such as planning ground rules and constraints, flight rules, workstation limits) in a standard format that supports consistency, traceability, analysis and efficient updates when changes occur during the plan and train phases of human spaceflight missions. It is currently under development.

It is anticipated that vendor-provided data sets can be used as a starting point for the development of failure mode and effect information tools and data sets customized to MOD's needs. An example of data sets that can be reused comes from the Constellation program. In Constellation, the Ares project, the Orion project, and KSC Ground Operations each selected the commercial-off-the-shelf product TEAMS[5] to assess failures. TEAMS was employed by the Ares project in support of design phase and operational phase needs. The Ground Operations project indicated an interest in using the same tool and data in support of vehicle integration and test operations at KSC. And, Orion initially envisioned using TEAMS as an integral part of the flight software; however, resource constraints moved the capability from onboard to ground software. This shared approach to failure assessment means that a set of models for each vehicle could be developed once and then reused to support the diverse needs of each project. Because of the advantages of this approach,

[5] TEAMS (Testability Engineering and Maintenance System) is a product suite of Qualtech Systems Inc. (www.teamqsi.com), QSI. It consists of TEAMS-Designer, a model design/building tool; TEAMS-RT, the real-time diagnostic tool; and TEAMS-RDS (Remote Distributed Server), a server-based approach to running multiple versions of TEAMS-RT remotely that includes the infrastructure to manage and store data, call the appropriate TEAMS-RT client with its corresponding data, and post the corresponding diagnosis to the appropriate client.

future spaceflight operations projects may follow a similar concept of standard data sets and possibly even standard display formats.

Through ACAWS-developed technology, MOD can also benefit from the direct reuse of tools, techniques, and data developed by peer projects. However, vendor-provided data sets will likely provide a different level of detail, formatting, and usability from that required to support failure assessment activities of analysts, instructors, and flight controllers. Initial assessment indicates the following technology needs are not satisfied with currently available products and technologies:

- Data product conversion capabilities that enable use of engineering products to generate properly formatted (to MOD layout standards) operations products.
- Rich authoring environment that allows operations personnel to augment or simplify engineering-grade data sets to meet operational needs.
- Operational impact assessment prioritization and presentation capabilities supporting operations personnel needs.
- Ability to exercise standard operational e-procedures in system models for purposes of just-in-time plan and procedure validation.

ACAWS will utilize previous projects' data sets as examples during the development of technologies and operations concepts as a proof of concept. In particular, we will use schematics from the Shuttle program and TEAMS models from the Constellation program. Technologies and operations concepts will be developed as generically as possible to accommodate failure assessment tools that may be selected for future spaceflight projects. However, some implementation will necessarily be specific to our example investigative data sets and will need to be modified to utilize other data sets; some necessary modifications may be quite involved.

3. Spaceflight Operations

ACAWS must support MOD during *plan – train – fly* activities during the strategic, tactical and execution phases of spaceflight operations. ACAWS also needs to support development phase analysis, pre-flight operations product development, flight crew and personnel training-simulation scripting, and real-time flight operations. It must support analysts, operators (i.e., flight controllers), instructors, and instructor- and operator-trainees. For current vehicles (Shuttle and ISS), the tasks that each role is responsible for are shown in Table 1. (The table provides background information to help the reader better understand mission operations; however, for future vehicles, these tasks may differ.) A single person may take on each role (excluding the crew function) and could result in over-lapping tasks. Although the initial version of ACAWS is focused on developing technologies to assist with MOD activities, similar information – perhaps at a different level of detail and in a different format – will be useful to the crew, especially for deep-space missions that will experience communication time delays. Thus, the crew role and associated tasks are included in the table. However, the operations concept for collaboration between crew and ground

operators and distribution of tasks under communication delays will need to be developed further. This expansion of ACAWS is outside the scope of the current document.

Table 1: Spaceflight operations roles and associated tasks.

Role	Tasks

Instructor	Develop training scenarios/malfunction case(s)
	Mentor trainee(s)
	Develop training material
	Support procedure development/verification
	Develop training concepts
	Conduct training (generic/flight specific)
	• Training media
	  o Tabletop/classroom
	  o Part-task trainer (PTT)
	  o Full-task trainer (FTT)
	  o Desktop/remote
	  o On-board
	• Instruct Space Flight Resource Management (SFRM)[6]
	• Proficiency/refresher
	• External customer
	• Perform student debrief
	• Perform student evaluation
	Develop simulator requirements/plans for upgrades
	Support boards/meetings
	Perform administrative tasks

Trainee (Instructor or Operator)	Use aforementioned training media to acquire knowledge
	Acquire knowledge (train)
	• Instructional skills
	• Console operations
	• Tools and applications
	• On-console communication system
	• SFRM
	• Procedures and nomenclature
	• Nominal vehicle systems operation
	• Off-nominal vehicle systems operation
	  o Troubleshoot anomalies (perform what-if analysis)
	  o Generate workaround plans
	Perform administrative tasks

Operator	Procedure development/verification Mission timeline development • Replan/workaround

[6] Space Flight Resource Management (SFRM) is similar to aircraft Crew Resource Management (CRM).

	Monitor vehicle systems Diagnose/isolate anomaly Troubleshoot anomaly Recover from anomaly • Replan/workaround Develop operations products Support boards/meetings Perform administrative tasks
Analyst	Predict flight system performance • System output capability reports • Consumables usage and systems availability • Resource margins • Operational limits Support boards/meetings Deliver products in support of flight production process Perform administrative tasks

Crew	Acquire knowledge (train)
	• Nominal vehicle systems operation
	• Off-nominal vehicle systems operation
	  o Troubleshoot anomalies (perform what-if analysis)
	  o Generate workaround plans
	Monitor vehicle systems
	Diagnose/isolate anomaly
	Troubleshoot anomaly
	Recover from anomaly
	• Replan/workaround
	Support boards/meetings

Some example MOD specific needs that ACAWS must support include the following:

- Reuse engineering products to generate products customized for operations use.
- View vehicle and system architecture in familiar graphical and tabular formats.[7]
- View vehicle and system architecture at multiple levels of abstraction (from a wide view of all vehicle systems down to specific subsystems, Line Replaceable Units (LRUs), or Orbital Replacement Units (ORUs)) while retaining situational awareness.
- View telemetry-driven indicators superimposed onto vehicle and system architecture. Telemetry values can be downlinked from the vehicle, from a simulation, or requested from archive[8].

[7] The system of interest may be a software system with software functions as the subsystems.

- Set model of systems in specific configurations and view resulting system architecture. For example, if a switch is set to the "off" position, the path to components beyond that switch is blocked (barring other paths) and can be displayed as inactive.
- Set model of systems in specific configurations (e.g., power status on/off, switch on/off, flight computer modes) and then investigate operational capabilities of the as-configured model.
- Insert malfunction scenarios (by setting specific configurations and telemetry values manually [entered interactively or from a file]) and investigate possible failures.
- Receive possible root cause failures for a given set of indicators, either telemetry-driven or MOD-initialized.
- Assign severity and priority properties to given failures and impacts.
- Receive mission impact of diagnosed failure.
- Receive mission impact of interactively injected failure. Impacts include both mission specific objectives that can change from flight to flight as well as those objectives that do not change from mission to mission and are considered generic.
- View failure impact reports organized by system, severity, or response priority.
- View proper procedural response for each failure and/or failure impact. The procedural response indicates the corrective actions – typically execution of a specific procedure – required of each operator.
- "Run" a set of activities/procedures in the model given a set of defined initial conditions. This enables on-the-fly desktop validation of procedures and timelines after unanticipated changes in vehicle configuration. To clarify, some subset of the steps in any procedure necessarily change the state of the system. By identifying those steps, one can change the configuration of the system model to match what would be expected when running the procedure.

4. Development Approach

ACAWS will be developed in three main phases. Within each phase, development will follow an agile development approach in which the requirements will evolve through collaboration with the users – the vehicle operators. We will begin the process with the initial list of requirements below which will be pruned and expanded to suit the needs of the users as we learn more about the problem and its solution. The three phases of ACAWS will each lead to a prototype system, simply named A, AB, and ABC. Each prototype is built onto its predecessor. A description of each prototype and its general capabilities are listed in Table 2.

Table 2: ACAWS development phases – capabilities and schedule.

[8] Archive system in use today is ODRC.

Capability	Specific Needs	Deliverables	Delivery Date
A	**Development Phase Analysis**		**7/2011**
Reuse engineering products to generate products customized for operations use.	• Ability to import and reconfigure engineering data sets to conform to operations standards for graphical layout and data presentation. • Fusion of graphical elements and layouts with telemetry-driven indicators to indicate system configuration. • Ability to view vehicle and system depictions at multiple levels of abstraction (from a wide view of all vehicle systems down to inspection of specific subsystems or Line Replaceable Units) while retaining situational awareness.	Requirements document, Dev A.	3/2010
		ConOps: Sketch of interface and user interaction, Dev A.	7/2010
		Prototype tool, Dev A.	7/2011
B	**Pre-Flight Operations Product Development**		**3/2013**
Customize authoring and inspection of vendor provided TEAMS data sets into operations-relevant system models.	• Easy inspection of system architecture (in familiar graphical and tabular formats). • Ability to investigate the impacts of selected single or multiple user-specified failure conditions. • Ability to investigate the operational capabilities of systems in specific configurations (example: availability of system functions based on on/off power status, flight computer mode, etc.). • Ability to assign – manually or automatically - severity and priority properties to given failures and impacts.	Expanded requirements document, Dev AB.	11/2011
		ConOps: Sketch of interface and user interaction, Dev AB.	3/2012
		Expanded prototype tool, Dev AB.	3/2013
C	**Real-Time and Training-Simulation Scripting Support**		**9/2016**
Determine full impact of a given set of spacecraft configurations and/or failures on crew/vehicle safety and mission success during real-time operations and simulation script development.	• Ability to initialize or update a system model based on telemetry values downlinked from the vehicle or a simulation. This may enhance the operator's situational awareness while reducing total training and certification costs for that operator. • User-friendly manipulation of system models to set system configurations and insert malfunction scenarios. • Easy inspection of failure impact reports with a variety of inspection options (organized by system, severity, or response priority). • Indication of proper procedural response for each failure and/or failure impact (indicating the corrective actions – typically execution of a specific procedure – required of each operator). • Ability to "run" a set of activities/procedures in the model given a set of defined initial conditions. This enables on-the-fly desktop validation of procedures and timelines after unanticipated changes in vehicle configuration.	Expanded requirements document, Dev ABC.	1/2014
		ConOps: Sketch of interface and user interaction, Dev ABC.	10/2014
		Expanded prototype tool, Dev ABC.	9/2016

5. Functional Requirements

Based on the desired capabilities and specific needs listed in Table 2, the initial user requirements for ACAWS are listed in the following tables. These requirements do not

represent software specifications. Rather, they represent the user's needs (user stories[9]) for an ACAWS system.

The requirements are color coded to facilitate recognition, with prototype "A" requirements in black, prototype "AB" in orange, prototype "ABC" in magenta, and "F" – possible future follow-up requirement candidates – in green.

The "Add in" column represents the prototype that that capability will be added in. Candidate requirements for follow-up work are shown with "Add in" of "F."

The "Priority" column represents the importance of the requirement to the user and has a value of 1 to 5 with 1 for highest priority items and 5 for lowest priority items. Higher priority requirements will be implemented first.

Sorting the following tables by "Req. #" provides a logical flow of requirements. Sorting by "Add in" provides the requirements by development phase. Sorting by "Priority" provides the requirements rank. Sorting by the "Add in" column and then by "Priority" provides the requirements in the order that they will be implemented using the agile development process.

References to "model" and "data set" are used interchangeably in the following requirements and mean "a representation of something." Thus, in the case of diagnostic models, we are referring to a model that explicitly models hierarchical interconnections, captures relationship between faults and observable effects ("tests"), captures redundancies and how models change with redundancy, and support multiple configurations of a system. In the case of an engineering model, we are referring to a schematic representation of the system. In the case of an ACAWS model, we are referring to the merged representation of the information from a schematic and the information in a diagnostic model. To avoid confusion with the underlying diagnostic or engineering model, we often refer to ACAWS models as ACAWS data sets.

5.1 General

Req. #	Requirement & *Rationale*	Add in	Priority
G1	Ability to support operators to plan, train for, and fly a mission.	A, B, C, F	1
G11	Ability to reuse diagnostic models that provide interconnections between components, capture the relationship between faults and fault detectors (health status indicators), and support multiple configurations of a system.[10,11]	A	1

[9] In the agile software development methodology, a user story is an informal method for users to influence the software's development. To formalize a user story into a formal software specification requires a corresponding acceptance test procedure.
[10] In the investigative prototypes, ACAWS shall reuse TEAMS models. Significant changes may be necessary to switch to a different tool.

G20	Ability to utilize MCT when feasible. *This will enable an integrated MCC tool set and allow ACAWS to reuse standardized infrastructure.*	A	1
G3	Ability to support multiple operators independently of each other. [8]	A	1
G4	Ability to have a consistent user interface for planning, training, and flight. That is, the UI should look the same at the flight console, at training facilities, and at a user's desktop.	A	1
G5	Ability to use telemetry values (as the source of input data) generated by a vehicle, generated by a simulator, or played back from an archive. [12]	A, F	1
G8	Ability to run under Linux OS. a. (TBR) Which variant of Linux? Red Hat?	A	1
G17	Ability to save ACAWS data sets generated from engineering and diagnostic models.	A	2
G18	Ability to load an existing ACAWS data set.	A	2
G13	Ability to select the graphical representation of a certain type of component (e.g., pump, hydraulic line) from a pre-specified palette. If a representation is not selected, reuse the component's graphical representation as it was imported from the engineering model.	A	3
G6	The source of telemetry values – either from a vehicle, from a simulator, or from an archive – must be apparent at all times.	A	3
G7	Ability to set or change the source of telemetry values - either from a vehicle, from a simulator, or from an archive – without requiring extra technical skills.	A	3
G12	Ability to display models in MOD graphical format standards.	A	4
G15	Ability to freeze an ACAWS data set when connected to real-time telemetry data. *This would prevent inadvertent modifications of the data set during a flight.*	B	1
G16	Ability to modify ACAWS data sets independently of concurrent ACAWS use by other operators. *This will allow an operator to explore changes to a data set without affecting the operations of other users.*	B	1
G18-2	Ability to save ACAWS data set modifications.	B	2
G14	Ability to modify ACAWS data sets ("models") without requiring extra technical skills.[13] *Rationale: systems may undergo changes for the first several*	B	3

[11] Diagnostic trees, also considered a model, may not provide enough of the needed information.
[12] In the investigative prototypes, ACAWS shall use archived telemetry values.
[13] Note that the modified models will then need to go through a VV&A process. This process is out of the scope of the ACAWS project. At a minimum, modified models will need to pass regression tests.

	flights and even throughout the lifetime of the vehicle. The model interface needs to be simple and the models easily modified by the operators so as to not add excessive overhead to a discipline's staffing needs.		
G2	Ability to support multiple operators simultaneously. *This allows multiple operators to start an ACAWS application, specifying the model(s) they want to load, telemetry stream to connect to, configuration of the model, etc.* [14]	F	
G9	Ability to run under Windows OS.	F	
G10	Ability to run under Mac OS.	F	
G21	Ability to verify, validate, and accredit (VV&A) the translation from schematics and diagnostic models to ACAWS models.	F	
G22	Ability to verify, validate, and accredit (VV&A) models modified in ACAWS.	F	

5.2 System Monitoring

Req. #	Requirement & *Rationale*	Add in	Priority
M1	Ability to obtain a health status of modeled systems, e.g., components are good, bad, suspect, unknown, degraded, etc.	A	1
M4	Ability to view a physical schematic of a system.	A	1
M5	Ability to view a functional model of system.	A	1
M6	Ability to view health status onto schematic, depicting component's bad, suspect, good, or unknown status.	A	1
M8	Ability to automatically or with little effort link diagnostic model components to corresponding schematic model components to corresponding ACAWS model components.	A	1
M18	Ability to view telemetry values superimposed onto schematic.	A	1
M28	Ability to display pertinent information in a generally shallow hierarchy. *Showing more information on one level rather than forcing navigation to get to details reduces cognitive load of remembering information from multiple sources necessary for making decisions.*	A	1
M13	Ability to show spatial information (e.g., physical location of heaters).	A	2
M17	Ability to visually distinguish health and (in)active state of ACAWS components[15] (e.g., green or white when that item is active/available, grayed-out or iconified when that item is not pertinent).	A	2
M20	Ability to set annunciation limits for telemetry per	A	2

[14] Similar to multiple users starting up Microsoft Word.
[15] The word *components* is used generically to represent systems, subsystems, LRUs, ORUs, or individual components.

	operational phase. *Telemetry limits can be used to support both health status computations and to provide alerts to the operator.*		
M22	Ability to navigate up/down model hierarchy in which the focus is only on the detailed view, without the surrounding context.	A	2
M23	Ability to visually determine location (level of depth) in the hierarchy during navigation.[16]	A	2
M26	Ability to pan over an ACAWS model. views.	A	2
M19	Ability to visually distinguish telemetry exceedances, using operational standards, e.g., for color coding.	A	3
M24	Ability to navigate down to a lower level on part of the system architecture while maintaining surrounding structure. That is, ability to expand an area of interest in situ but retain the surrounding architecture in its current expansion state, like holding up a detail-exposing magnifying glass over a part of the architecture and being able to not just see it bigger but in more detail.	A	3
M29	Ability to overlay multiple selected layers (e.g., electrical links and hydraulic links of a system) on a single display.	A	3
M31	Ability to display multiple windows simultaneously. *This will allow user to display detailed views of multiple systems, a less detailed view of the whole system, and simultaneously view any reference materials associated with the model.* (a) Ability to move or overlay the windows. (b) Ability to automatically juxtapose the active windows.	A	3
M50	Ability to obtain health status of ACAWS itself.	A	3
M25	Ability to zoom (in/out) an ACAWS model. (a) (TBR) Specify whether discrete zoom levels, continuous zoom, or "rubber-banded" zoom. Enumerate levels of zoom available.	A	4
M21	Ability to view models and health status at different hierarchy levels (system, subsystem, LRU, ORU, component, failure mode).	A	5
M3	Ability to visually confirm which systems, subsystems, LRUs, or ORUs have a diagnostic model associated with them.	A	5
M9	Ability to automatically or with little effort update diagnostic model.	B	1
M10	Ability to clearly distinguish modifications of a ACAWS data set. *This will facilitate VV&A.*	B	1
M11	Ability to automatically or with little effort maintain consistency between ACAWS data sets and diagnostic models.	B	1
M14	Ability to *manually* set model of systems in specific	B	1

[16] Example implementation: show "breadcrumb" of hierarchy location.

	configurations and view resulting system architecture. For example, if a switch is set to the "off" position, the path to components beyond that switch is blocked (barring other paths) and can be displayed as inactive.		
M15	Ability to *automatically (based on telemetry)* set model of systems in specific configurations and view resulting system architecture. For example, ground-only-relevant components of a model are shown as inactive after vehicle lifts off. *This reduces clutter and enhances ability to detect important details.*	B	1
M27	Ability to view related ACAWS models by selecting its reference item in the present model. (That is, if you click on a "connecting" area, the other model is displayed, either [operator selectable] replacing the current schematic or in a separate window.)	B	1
M35	Ability to locate and navigate to part of model showing "suspect" and "bad" components.	B	1
M36	Ability to view tests[16] at a test point in a separate window. This includes viewing both the test itself (i.e., which measurement is being evaluated against which value) and the current result of the test.	B	1
M2	Ability to view system status in MOD tabular format, to resemble an MSK display.	B	2
M12	Ability to link external documents to an item (e.g., photograph).	B	2
M16	Ability to view both active and inactive portions of system architecture.	B	2
M37	Ability to view[17] connecting path between two selected items, or if no connection exists, receive a message to that effect.	B	2
M38	Ability to highlight a propagation path from a component onward in given configuration and/or in various system modes.	B	2
M34	Ability to search ACAWS models by subsystem, LRU, ORU, component, unique-identifier, suspect/bad components, or line labels, and view the search results graphically. Jump to searched place.	B	3
M41	Ability to save viewing preferences (saving selected options for what information is overlaid/hidden, whether windows are overlapped or tiled, whether new schematics open in own window, etc.).	B	3
M42	Ability to recall a user's saved viewing preferences.	B	3
M43	Ability to edit a user's viewing preferences.	B	3

[17] If connection exists, it can be highlighted or shown by minimizing everything in between.

M44	Ability to save viewing configuration (saving which data sets are shown, how those windows are arranged, etc.)	B	3
M45	Ability to recall a saved configuration.	B	3
M7	Ability to move components on the display. *This allows the user to easily juxtapose areas of interest.*	B	4
M30	Ability to hide/show various overlaid information (e.g., observation points, observable effects, [18] aspects related to physical & functional (e.g., electrical) structure, telemetry values, …)	B	5
M39	Ability to organize and index ACAWS data sets.	B	5
M40	Ability to search for a data set in ACAWS data set database.	B	5
M52	Ability to merge multiple schematics into one ACAWS schematic. *Requirement M27 provides the operator the ability to follow a link to a related model from the currently viewed model. This requirement provides for a different operations concept – one of embedding related models into an integrated model.*	B	6
M49	Ability to view failure messages prioritized by how quickly a response is necessary, propagation timing of failure to other systems, and mission impact, or type of root cause message.	C	4
M32	Ability to interact with ACAWS via speech input. (For example, "bring up schematic for EPS.") *Use of speech in MCC environment may be impractical. However, speech may be useful for space flight crews, especially if they are suited (ascent/entry or EVA).*	F	
M33	Ability to view a plot of a unique-identifier by selecting the unique-identifier on the schematic. (Setting up and screen placement of a plot shall be saved as part of a user defined preference.)	F	
M46	Ability to receive output via speech. (For example, failures requiring quick response are annunciated verbally as well as visually).	F	
M47	Ability to view a time series plot of the diagnoses.	F	
M48	Ability to specify new tests directly on the ACAWS model.	F	
M51	Ability to aggregate information from different areas into a single user-configured area.	F	

5.3 Failure

ACAWS shall replicate or extend CRANS functionality.

[18] Tests are the observable effects. In the investigative prototype based on TEAMS, these would be test results being sent to TEAMS. "Observation points" would be "test points", points where multiple tests may be attached to the model. We use "tests" and "observable effects" interchangeably. Similarly, we use "observation points" and "test points" interchangeably.

Req. #	Requirement & *Rationale*	Add in	Priority
F5	Ability to view the suspect components in probabilistic order (most probable to least probable).	A	2
F6	Ability to identify if more than one component is suspect.	A	2
F1	Ability to view root cause failure for set of failed items (set through telemetry, interactively, or through configuration file) to the extent that the reference materials capture the root cause.	B	1
F2	Ability to view common failure for set of selected items (interactively "tagged" item or via telemetry). (Replicate and extend CRANS "commonality" capability.)	B	1
F3	Ability to receive rationale (on demand, graphically or textually) of why a certain failure was selected as the "bad" (failed) one.	B	3
F4	Ability to view failure modes as groups that make sense. For example, LCC applies to leaks anywhere in the system. All failure modes that might be classified as a leak could be grouped into a single failure type that ACAWS would present.	B	4
F7	Ability to support troubleshooting in cases when more than one component is suspect to isolate to a minimal ambiguity group based on the available test suite.	F	1

5.4 Impact

Req. #	Requirement & *Rationale*	Add in	Priority
I1	Ability to identify all items that would fail for a set of selected items (interactively "tagged items" or via telemetry). (Replicate and extend CRANS "fail result" capability.)	C	1
I10	Ability to view critical equipment lost upon failure. (a) (TBR) Can we get list of critical equipment by phase from another tool? Should it come from other tool or should we expect to put a static list in ACAWS? Sotirios will check.	C	1
I2	Ability to view all items that would fail for a specified unique-identifier.	C	1
I6	Ability to view all items that can propagate failure to interactively selected item(s) or telemetry-failed item(s).	C	1
I7	Ability to view unique-identifiers that can cause an item to fail.	C	1
I3	Ability to perform "what if" analysis for subsequent failures. *This would allow operator to determine next worst failure.*	C	2
I5	The nature of the currently viewed information – live/simulated/archived telemetered data vs. manually selected "what if" configuration – must be apparent at all times.	C	2
I4	ACAWS shall distinguish representation of interactively and telemetry-set failed items.	C	3

I8	Ability to suppress previous failures to make new failures more evident.	C	3
I11	Ability to view impact of failure to mission. (a) Ability to determine whether failures affect ongoing or upcoming operations. (Identified on the timeline, like what we saw with A4O?) Ability to view when mission is one failure away (fail critical) from being unable to perform a scheduled operation.	C	3
I12	Ability to view list of flight rules and constraints that are impacted due to failure(s).[19]	C	3
I9	Ability to associate and view timing of failure propagation.	C	4

5.5 Recovery and Workaround

Req. #	Requirement & *Rationale*	Add in	Priority
W2	Ability to view corresponding recovery procedure for a given failure.	C	2
W1	Ability to list the recovery procedures associated with a failure.[20]	C	3
W3	Ability to recommend a recovery response to multiple failures based on severity of impact.[19]	C	3
W4	Ability to interact with a Flight Note (FN) system. *A FN can be used to alter procedures, document status reports from the Flight Control Team (FCT), disseminate information among the flight control community, and update summaries to send to the crew. FNs are used to add words to the daily summary/execute package for the crew concerning operations and related questions and answers.*	F	1
W5	Ability to interact with an anomaly reporting system.	F	1
W6	Ability to interact with a reporting system (like a CHIT). [*A CHIT report is used by NASA flight support agencies to make official inquiries for information from other groups during a mission. Historically, CHITs were most often used by the FCT to request engineering and safety analysis from the MER.*]	F	1

6. Design

In the current state of mission operations, typically each tool has its own user interface and own data set, implements its own connection to a data source, and interacts only with either the user or posts its computations to ISP but does not have an API to interact with other tools. In a future state of mission operations enabled by ACAWS and the MCT tool, many tools will be able to receive data (telemetry, archived, or simulated) through services provided by MCT. By connecting via MCT, tools like ACAWS will be able to more easily

[19] ConFRM tool currently under development may provide this information. Prefer to integrate ACAWS with ConFRM rather than duplicating functionality.
[20] CxPASS tool currently under development may provide this information. Prefer to integrate ACAWS with CxPASS rather than duplicating functionality.

integrate with other tools, such as MSKView, ELOG, RT-Plot, ConFRM, etc. Within this scenario, ACAWS will primarily need to connect only to a diagnostic engine, diagnostic models, and any tools that are not yet connected to MCT. More importantly for the operator, all tools that connect via MCT will allow similar interaction capabilities, simplifying flight planning, training, and execution. Figure 7 depicts the two different approaches.

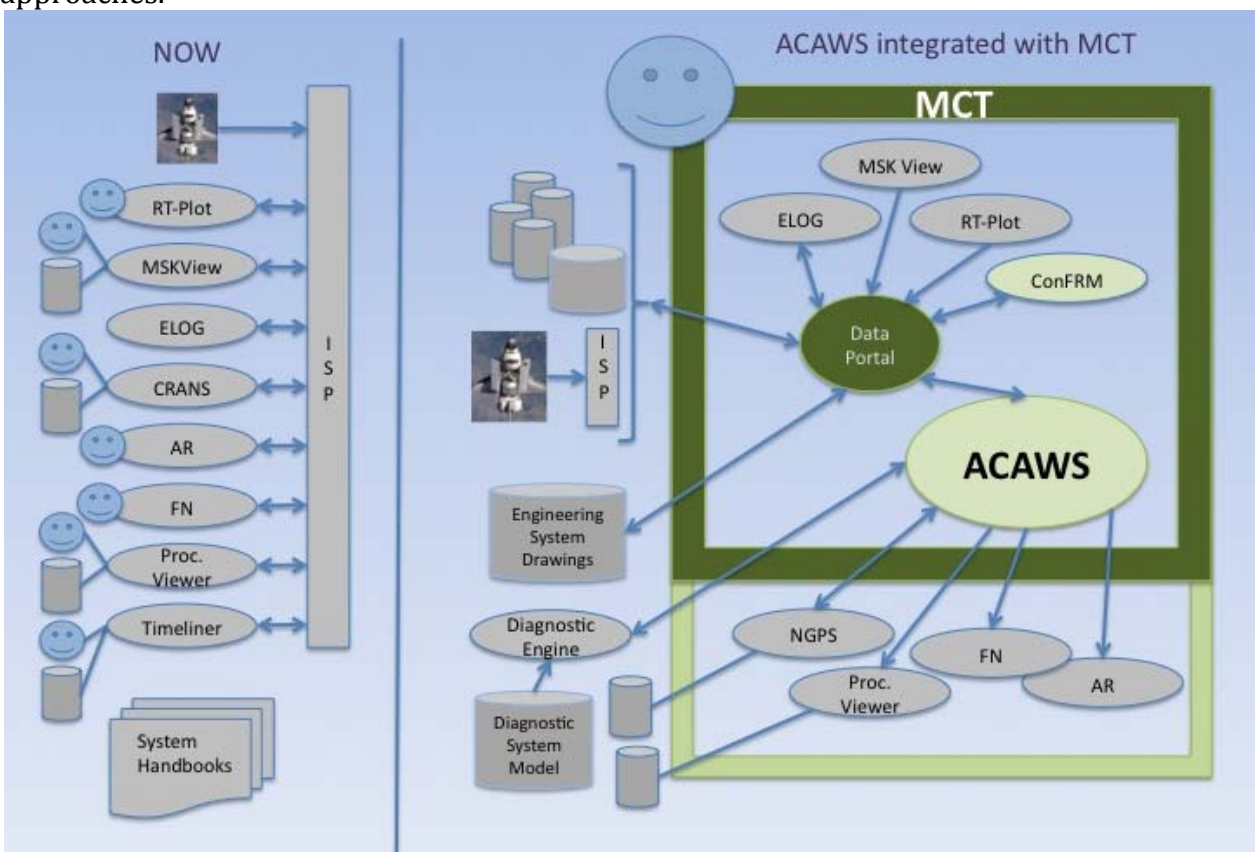


Figure 7: Two different views of mission operations state: now and with ACAWS integrated with MCT.

6.1 High-Level Architecture

ACAWS is made up of eight complementary modules, as shown in Figure 8. These can be implemented in one large process, with each module represented by a software code method/function, or they can be split into multiple processes with communication between processes through shared memory, a message bus, or a different approach. Such decisions are left for the implementation phase of the project and will be made by the developers.

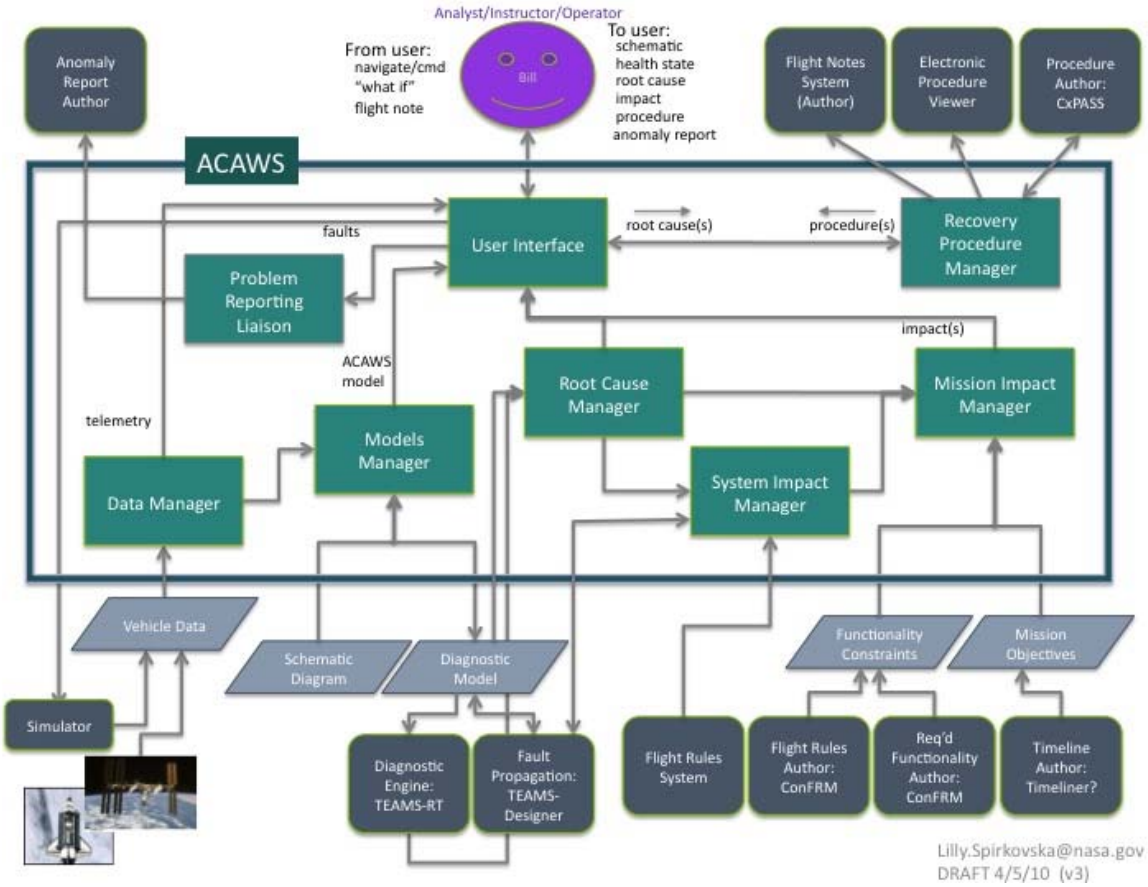


Figure 8: ACAWS high-level architecture.

The eight modules, with short descriptions of each, are the following:

- *Models Manager:* Concerned primarily with creating the ACAWS model by merging information from the schematic diagram and diagnostic model, providing the ability to modify the model and get those modifications propagated back to the diagnostic engine, and augmenting the models by enabling the operator to attach external documents to model components.
- *Root Cause Manager:* Responsible for determining the health status of the modeled system(s), determining the root cause of any out-of-limit telemetry observations, grouping and prioritizing root cause or fault messages, providing rationale on why a particular root cause explains the observations, and helping the operator troubleshoot in cases where multiple suspects are feasible.
- *System Impact Manager:* Identifies the impact (effect) of any failures (of the vehicle or injected by the operator for "what if" scenario exploration) to downstream components, determines if multiple components selected ("tagged") by the operator have a common cause failure, determines critical components that may be lost due

to a (actual or operator injected) failure, and provides the timing of failure propagation to downstream components.

- *Mission Impact Manager:* Determines the impact to the mission of components that have failed (initial and propagated downstream failures). Whereas the system impact manager looks at failures at a system level, the mission impact manager considers the expected mission activities, the functionality required to accomplish those activities, and the system capabilities required to obtain that functionality. Thus, the system impact manager provides information to the mission impact manager. Other systems, external to ACAWS, provide the remaining necessary information.
- *Recovery Procedure Manager:* Determines the appropriate procedure to run to recover from a failure. In addition to the automatic selection of a procedure, it allows the operator to obtain a list of possibly applicable procedures. When multiple failures occur, it prioritizes the order of procedures that should be executed. Finally, it coordinates with a flight notes system to enable the operator to insert additional notes or comments to a procedure. In the future, ACAWS can coordinate with the Automation for Operations (A4O) systems for automated procedure creation and execution.
- *Problem Reporting Liaison:* Coordinates with the problem reporting system to enable the operator to enter a problem report or a CHIT. ACAWS automatically fills in as much information as it can; the operator fills in other fields.
- *Data Manager:* Connects to the telemetry stream from the vehicle, from a simulator, or from an archived data file and feeds that data stream to other ACAWS modules. The data manager also coordinates connection to MCT where that is feasible and desirable for ACAWS.
- *User Interface:* The collection of functionality to allow the user to view with the models, view health status of systems, overlay additional information onto models (e.g., observations, telemetry, …), navigate through the models (e.g., pan, zoom, drill up/down hierarchy, to another model, …), search the models (for, e.g., a sensor, component, failed components, …), and view configuration and status information about the ACAWS system itself (vs. the system being examined).

Figure 9 shows a summary of the ACAWS subsystems and their capabilities and responsibilities.

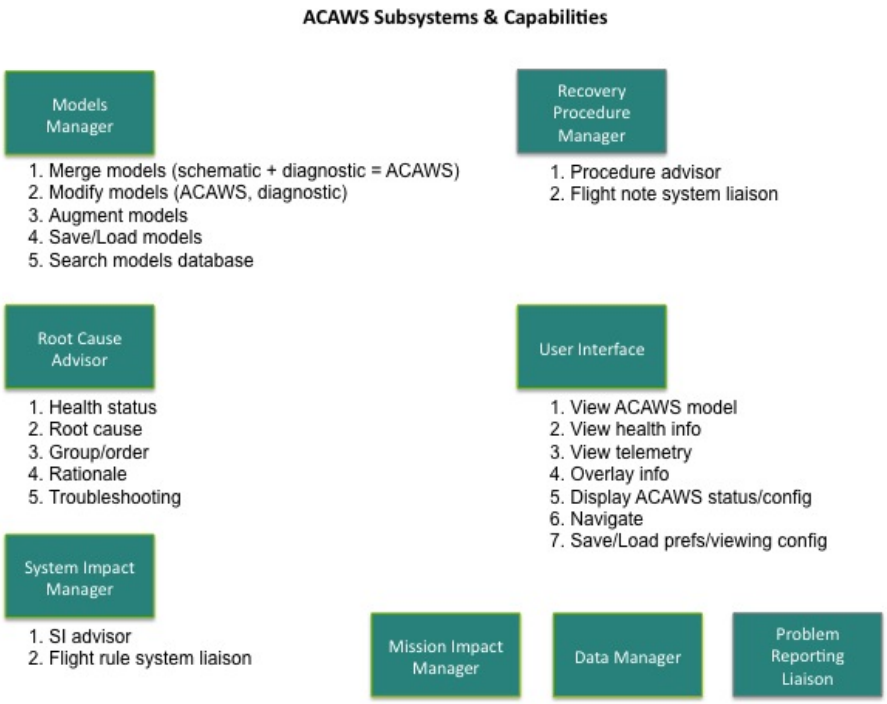


Figure 9: ACAWS subsystems and categories of capabilities.

6.2 Functional Requirements Distribution

The following tables reorganize the functional requirements of Section 5 into the ACAWS subsystems and categories shown in Figure 9. Note that the last column in each table has been renamed "Demo." Although the requirements were developed with three implementation phases, due to funding program priorities, we are developing a concept demonstration version that covers functionality from all phase. Some of these functions will be implemented with code, some possibly coded, some implemented with "smoke and mirrors" (i.e., mocked up in concept but not fully implemented), some storyboarded, and some just mentioned during the presentation. The following table provides the key to how to interpret the demo column color-coding and column content. The demo column represents the group's consensus on how functionality will be demonstrated.

Table 3: Demonstration column color-coding and character key.

How demo'ed	Color	Description
Coded	Yellow	C
Possibly coded	Pale green	C?
Smoke & Mirror	Peach	SM
Storyboard	Lavender	S
Presented	White	P
Not demo'ed	Gray	N/A

Table 4: Generic requirements.

Req. #	Requirement & *Rationale*	Add in	Demo
G1	Ability to support operators to plan, train for, and fly a mission.	A, B, C, F	P
G3	Ability to support multiple operators independently of each other.[8]	A	P
G4	Ability to have a consistent user interface for planning, training, and flight. That is, the UI should look the same at the flight console, at training facilities, and at a user's desktop.	A	P
G8	Ability to run under Linux OS. a. (TBR) Which variant of Linux? Red Hat?	A	P
G2	Ability to support multiple operators simultaneously. *This allows multiple operators to start an ACAWS application, specifying the model(s) they want to load, telemetry stream to connect to, configuration of the model, etc.*[21]	F	P
G9	Ability to run under Windows OS.	F	P
G10	Ability to run under Mac OS.	F	P

Table 5: Model Manager -- Merge Models capability.

Req. #	Requirement & *Rationale*	Add in	Demo
G11	Ability to reuse diagnostic models that provide interconnections between components, capture the relationship between faults and fault detectors (health status indicators), and support multiple configurations of a system.[22,23]	A	C
G13	Ability to select the graphical representation of a certain type of component (e.g., pump, hydraulic line) from a pre-specified palette. If a representation is not selected, reuse the component's graphical representation as it was imported from the engineering model.	A	P
M8	Ability to automatically or with little effort link diagnostic model components to corresponding schematic model components to corresponding ACAWS model components.	A	C
M52	Ability to merge multiple schematics into one ACAWS schematic. *Requirement M27 provides the operator the ability to follow a link to a related model from the currently viewed model. This requirement provides for a different operations concept – one of embedding related models into an integrated model.*	B	S

[21] Similar to multiple users starting up Microsoft Word.
[22] In the investigative prototypes, ACAWS shall reuse TEAMS models. Significant changes may be necessary to switch to a different tool.
[23] Diagnostic trees, also considered a model, may not provide enough of the needed information.

Table 6: Model Manager -- Modify Models capability.

Req. #	Requirement & *Rationale*	Add in	Demo
G14	Ability to modify ACAWS data sets ("models") without requiring extra technical skills.[24] *Rationale: systems may undergo changes for the first several flights and even throughout the lifetime of the vehicle. The model interface needs to be simple and the models easily modified by the operators so as to not add excessive overhead to a discipline's staffing needs.*	B	P
M9	Ability to automatically or with little effort update diagnostic model.	B	P
M10	Ability to clearly distinguish modifications of a ACAWS data set. *This will facilitate VV&A.*	B	P
M11	Ability to automatically or with little effort maintain consistency between ACAWS data sets and diagnostic models.	B	P
M7	Ability to move components on the display. *This allows the user to easily juxtapose areas of interest.*	B	P
G15	Ability to freeze an ACAWS data set when connected to real-time telemetry data. *This would prevent inadvertent modifications of the data set during a flight.*	B	P
G16	Ability to modify ACAWS data sets independently of concurrent ACAWS use by other operators. *This will allow an operator to explore changes to a data set without affecting the operations of other users.*	B	P
G21	Ability to verify, validate, and accredit (VV&A) the translation from schematics and diagnostic models to ACAWS models.	F	P
G22	Ability to verify, validate, and accredit (VV&A) models modified in ACAWS.	F	P

Table 7: Model Manager -- Augment Models capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M12	Ability to link external documents to an item (e.g., photograph, specifications, …).	B	C

Table 8: Model Manager -- Save/Load Models capability.

Req. #	Requirement & *Rationale*	Add in	Demo
G17	Ability to save ACAWS data sets generated from engineering and diagnostic models.	A	C
G18	Ability to load an existing ACAWS data set.	A	C
G18-2	Ability to save ACAWS data set modifications.	B	P

[24] Note that the modified models will then need to go through a VV&A process. This process is out of the scope of the ACAWS project. At a minimum, modified models will need to pass regression tests.

Table 9: Model Manager -- Search Models Database capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M39	Ability to organize and index ACAWS data sets.	B	P
M40	Ability to search for a data set in ACAWS data set repository.	B	P

Table 10: Root Cause Manager -- Health Status capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M1	Ability to obtain a health status of modeled systems, e.g., components are good, bad, suspect, unknown, degraded, etc.	A	C
M21-2	Ability to obtain health status at different hierarchy levels (system, subsystem, LRU, ORU, component, failure mode).	A	C

Table 11: Root Cause Manager -- Root Cause capability.

Req. #	Requirement & *Rationale*	Add in	Demo
F1	Ability to view root cause failure for set of failed items (set through telemetry, interactively, or through configuration file) to the extent that the reference materials capture the root cause.	B	C

Table 12: Root Cause Manager -- Group/Order capability.

Req. #	Requirement & *Rationale*	Add in	Demo
F5	Ability to view the suspect components in probabilistic order (most probable to least probable).	A	SM
M49	Ability to view failure messages prioritized by how quickly a response is necessary, propagation timing of failure to other systems, and mission impact, or type of root cause message.	C	P
F4	Ability to view failure modes as groups that make sense. For example, LCC applies to leaks anywhere in the system. All failure modes that might be classified as a leak could be grouped into a single failure type that ACAWS would present.	B	S

Table 13: Root Cause Manager -- Rationale capability.

Req. #	Requirement & *Rationale*	Add in	Demo
F3	Ability to receive rationale (on demand, graphically or textually) of why a certain failure was selected as the "bad" (failed) one.	B	S

Table 14: Root Cause Manager -- Troubleshooting capability.

Req. #	Requirement & *Rationale*	Add in	Demo
F6	Ability to identify if more than one component is suspect.	A	C
F7	Ability to support troubleshooting in cases when more than	F	S

	one component is suspect to isolate to a minimal ambiguity group based on the available test suite.		

Table 15: System Impact Manager – System Impact Advisor capability.

Req. #	Requirement & *Rationale*	Add in	Demo
I1	Ability to identify all items that would fail for a set of selected items (interactively "tagged items" or via telemetry). (Replicate and extend CRANS "fail result" capability.)	C	C?
F2	Ability to view common failure for set of selected items (interactively "tagged" item or via telemetry). (Replicate and extend CRANS "commonality" capability.)	B	C?
I6	Ability to view all items that can propagate failure to interactively selected item(s) or telemetry-failed item(s).	C	C?
M38	Ability to highlight a propagation path from a component onward in given configuration and/or in various system modes.	B	C?
I2	Ability to view all items that would fail for a specified unique-identifier.	C	C
I7	Ability to view unique-identifiers that can cause an item to fail.	C	C
I9	Ability to predict and view timing of failure propagation.	C	C?
I10	Ability to view critical equipment lost upon failure. (a) (TBR) Can we get list of critical equipment by phase from another tool? Should it come from other tool or should we expect to put a static list in ACAWS? Sotirios will check.	C	C?
I3	Ability to perform "what if" analysis for subsequent failures. *This would allow operator to determine next worst failure.*	C	C?

Table 16: System Impact Manager -- Flight Rules Liaison capability.

Req. #	Requirement & *Rationale*	Add in	Demo
I12	Ability to view list of flight rules and constraints that are impacted due to failure(s).[25]	C	C?

Table 17: Mission Impact Manager.

Req. #	Requirement & *Rationale*	Add in	Demo
I11	Ability to view impact of failure to mission. (b) Ability to determine whether failures affect ongoing or upcoming operations. (Identified on the timeline, like what we saw with A4O?) Ability to view when mission is one failure away (fail critical) from being unable to perform a scheduled operation.	C	SM

[25] ConFRM tool currently under development may provide this information. Prefer to integrate ACAWS with ConFRM rather than duplicating functionality.

Table 18: Recovery Procedure Manager -- Procedure Advisor capability.

Req. #	Requirement & *Rationale*	Add in	Demo
W1	Ability to list the recovery procedures associated with a failure.[26]	C	C?
W2	Ability to view corresponding recovery procedure for a given failure.	C	C?
W3	Ability to recommend a recovery response to multiple failures based on severity of impact.[19]	C	S

Table 19: Recovery Procedure Manager -- Flight Note Liaison capability.

Req. #	Requirement & *Rationale*	Add in	Demo
W4	Ability to interact with a Flight Note (FN) system. *A FN can be used to alter procedures, document status reports from the Flight Control Team (FCT), disseminate information among the flight control community, and update summaries to send to the crew. FNs are used to add words to the daily summary/execute package for the crew concerning operations and related questions and answers.*	F	SM

Table 20: Problem Reporting Liaison.

Req. #	Requirement & *Rationale*	Add in	Demo
W5	Ability to interact with an anomaly reporting system. (ACAWS will autofill information in an AR that it has available to it & allow the operator to fill out the remainder.)	F	SM
W6	Ability to interact with a reporting system (like a CHIT). [*A CHIT report is used by NASA flight support agencies to make official inquiries for information from other groups during a mission. Historically, CHITs were most often used by the FCT to request engineering and safety analysis from the MER.*]	F	SM

Table 21: Data Manager.

Req. #	Requirement & *Rationale*	Add in	Demo
G5	Ability to use telemetry values (as the source of input data) generated by a vehicle, generated by a simulator, or played back from an archive.[27]	A, F	C
G7	Ability to set or change the source of telemetry values - either from a vehicle, from a simulator, or from an archive – without requiring extra technical skills.	A	P

[26] CxPASS tool currently under development may provide this information. Prefer to integrate ACAWS with CxPASS rather than duplicating functionality.
[27] In the investigative prototypes, ACAWS shall use archived telemetry values.

G20	Ability to utilize MCT when feasible. *This will enable an integrated MCC tool set and allow ACAWS to reuse standardized infrastructure.*	A	P

Table 22: User Interface -- View ACAWS Model capability.

Req. #	Requirement & *Rationale*	Add in	Demo
G12	Ability to display models in MOD graphical format standards.	A	C
M4	Ability to view a physical schematic of a system.	A	C
M5	Ability to view a functional model of system.	A	C
M13	Ability to show spatial information (e.g., physical location of heaters).	A	C
M16	Ability to view both active and inactive portions of system architecture.	B	S
M14	Ability to *manually* set model of systems in specific configurations and view resulting system architecture. For example, if a switch is set to the "off" position, the path to components beyond that switch is blocked (barring other paths) and can be displayed as inactive.	B	C?
M15	Ability to *automatically (based on telemetry)* set model of systems in specific configurations and view resulting system architecture. For example, ground-only-relevant components of a model are shown as inactive after vehicle lifts off. *This reduces clutter and enhances ability to detect important details.*	B	S

Table 23: User Interface -- View Health Information capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M6	Ability to view health status onto schematic, depicting component's bad, suspect, good, or unknown status.	A	C
M21	Ability to view models and health status at different hierarchy levels (system, subsystem, LRU, ORU, component, failure mode).	A	C
I9-2	Ability to view time stamps of actual failure propagation.	C	C
M17	Ability to visually distinguish health and (in)active state of ACAWS components[28] (e.g., green or white when that item is active/available, grayed-out or iconified when that item is not pertinent).	A	P
I8	Ability to suppress previous failures to make new failures more evident.	C	P
M2	Ability to view system status in MOD tabular format, to	B	N/A

[28] The word *components* is used generically to represent systems, subsystems, LRUs, ORUs, or individual components.

	resemble an MSK display.		

Table 24: User Interface -- View Telemetry capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M18	Ability to view telemetry values superimposed onto schematic.	A	C
M19	Ability to visually distinguish telemetry exceedances, using operational standards, e.g., for color coding.	A	C
M20	Ability to set annunciation limits for telemetry per operational phase. *Telemetry limits can be used to support both health status computations and to provide alerts to the operator.*	A	P

Table 25: User Interface -- Overlay Information capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M28	Ability to display pertinent information in a generally shallow hierarchy. *Showing more information on one level rather than forcing navigation to get to details reduces cognitive load of remembering information from multiple sources necessary for making decisions.*	A	C
M29	Ability to overlay multiple selected layers (e.g., electrical links and hydraulic links of a system) on a single display.	A	C
M30	Ability to hide/show various overlaid information (e.g., observation points, observable effects,[29] aspects related to physical & functional (e.g., electrical) structure, telemetry values, …)	B	C
M36	Ability to view tests[16] at a test point in a separate window. This includes viewing both the test itself (i.e., which measurement is being evaluated against which value) and the current result of the test.	B	C
M12-2	Ability to view external documents attached to an item.	A	C
M37	Ability to view[30] connecting path between two selected items, or if no connection exists, receive a message to that effect.	B	C
M33	Ability to view a plot of a unique-identifier by selecting the unique-identifier on the schematic. (Setting up and screen placement of a plot shall be saved as part of a user defined preference.)	F	SM

[29] Tests are the observable effects. In the investigative prototype based on TEAMS, these would be test results being sent to TEAMS. "Observation points" would be "test points", points where multiple tests may be attached to the model. We use "tests" and "observable effects" interchangeably. Similarly, we use "observation points" and "test points" interchangeably.

[30] If connection exists, it can be highlighted or shown by minimizing everything in between.

M47	Ability to view a time series plot of the diagnoses.	F	SM

Table 26: User Interface -- Display ACAWS Status/Configuration capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M50	Ability to obtain health status of ACAWS itself.	A	P
M3	Ability to visually confirm which systems, subsystems, LRUs, or ORUs have a diagnostic model associated with them.	A	P
G6	The source of telemetry values – either from a vehicle, from a simulator, or from an archive – must be apparent at all times.	A	SM
I5	The nature of the currently viewed information – live/simulated/archived telemetered data vs. manually selected "what if" configuration – must be apparent at all times.	C	N/A
I4	ACAWS shall distinguish representation of interactively and telemetry-set failed items.	C	N/A

Table 27: User Interface -- Navigation capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M26	Ability to pan over an ACAWS model. views.	A	C
M25	Ability to zoom (in/out) an ACAWS model. (b) (TBR) Specify whether discrete zoom levels, continuous zoom, or "rubber-banded" zoom. Enumerate levels of zoom available.	A	C
M22	Ability to navigate up/down model hierarchy in which the focus is only on the detailed view, without the surrounding context.	A	C
M24	Ability to navigate down to a lower level on part of the system architecture while maintaining surrounding structure. That is, ability to expand an area of interest in situ but retain the surrounding architecture in its current expansion state, like holding up a detail-exposing magnifying glass over a part of the architecture and being able to not just see it bigger but in more detail.	A	S
M23	Ability to visually determine location (level of depth) in the hierarchy during navigation.[31]	A	C
M35	Ability to locate and navigate to part of model showing "suspect" and "bad" components.	B	C
M27	Ability to view related ACAWS models by selecting its reference item in the present model. (That is, if you click on a "connecting" area, the other model is displayed, either [operator selectable] replacing the current schematic or in a separate window.)	B	C

[31] Example implementation: show "breadcrumb" of hierarchy location.

M31	Ability to display multiple windows simultaneously. *This will allow user to display detailed views of multiple systems, a less detailed view of the whole system, and simultaneously view any reference materials associated with the model.* (c) Ability to move or overlay the windows. (d) Ability to automatically juxtapose the active windows.	A	C
M34	Ability to search ACAWS models by subsystem, LRU, ORU, component, unique-identifier, suspect/bad components, or line labels, and view the search results graphically. Jump to searched place.	B	C

Table 28: User Interface -- Save/Load Preferences/Viewing Configurations capability.

Req. #	Requirement & *Rationale*	Add in	Demo
M44	Ability to save viewing configuration (saving which data sets are shown, how those windows are arranged, etc.)	B	P
M45	Ability to recall a saved configuration.	B	P
M41	Ability to save viewing preferences (saving selected options for what information is overlaid/hidden, whether windows are overlapped or tiled, whether new schematics open in own window, etc.).	B	P
M42	Ability to recall a user's saved viewing preferences.	B	P
M43	Ability to edit a user's viewing preferences.	B	P

Table 29: Unclassified requirements.

Req. #	Requirement & *Rationale*	Add in	Demo
M32	Ability to interact with ACAWS via speech input. (For example, "bring up schematic for EPS.") *Use of speech in MCC environment may be impractical. However, speech may be useful for space flight crews, especially if they are suited (ascent/entry or EVA).*	F	N/A
M46	Ability to receive output via speech. (For example, failures requiring quick response are annunciated verbally as well as visually).	F	N/A
M48	Ability to specify new tests directly on the ACAWS model.	F	N/A
M51	Ability to aggregate information from different areas into a single user-configured area.	F	N/A

6.3 Demonstration System Approach

The requirements enumerated in the tables of the previous section have been grouped into the top-level functionality we will demonstrate and prioritized by their potential impact on the customer's perception of ACAWS. Figure 10 and Figure 11 show this information in two

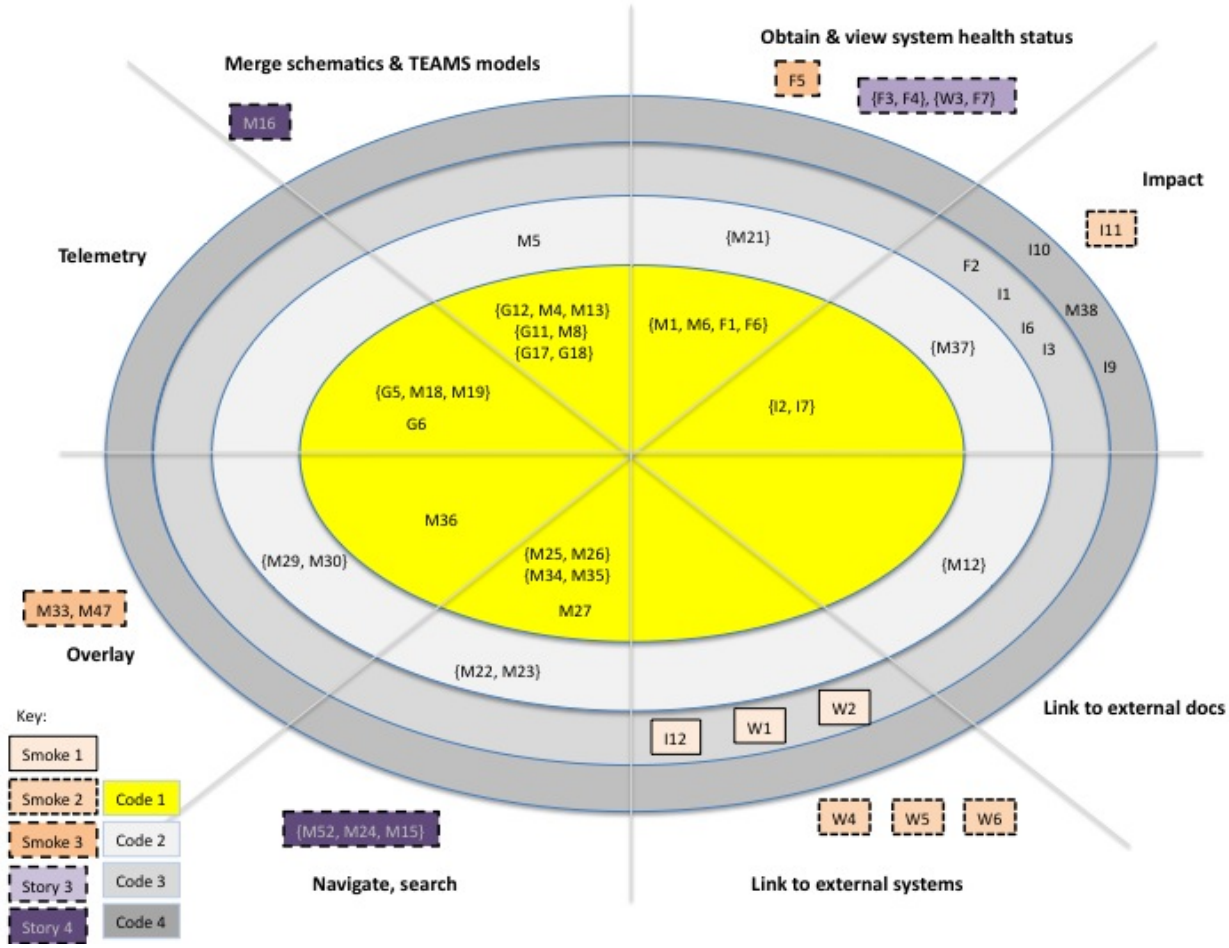


Figure 10: Grouping and prioritization of ACAWS demo capabilities. The eight groups are shown as pieces of the pie (i.e., oval), with the labels in bold on the outside of the pie. The highest priority items are in the inner yellow core. Priority of the outer rings decreases as the distances from the center increases. The priority of mocked-up items ("smoke" for "smoke and mirrors" implementation approach) and storyboarded ("story") items is color-coded and distinguished by the type of line around each rectangle.

ways. The underlying groupings and prioritization are the same; only the presentation differs. The goal is to demonstrate at least priority 1 and priority 2 items, with priority 3 and 4 added as time allows. This goal applies to requirements that will be coded, those that will be mocked up ("smoke & mirrors", shown as "Smoke" in the figures), and storyboarded. Any requirements not shown in the two figures will be discussed during a demonstration presentation but will not be shown explicitly.

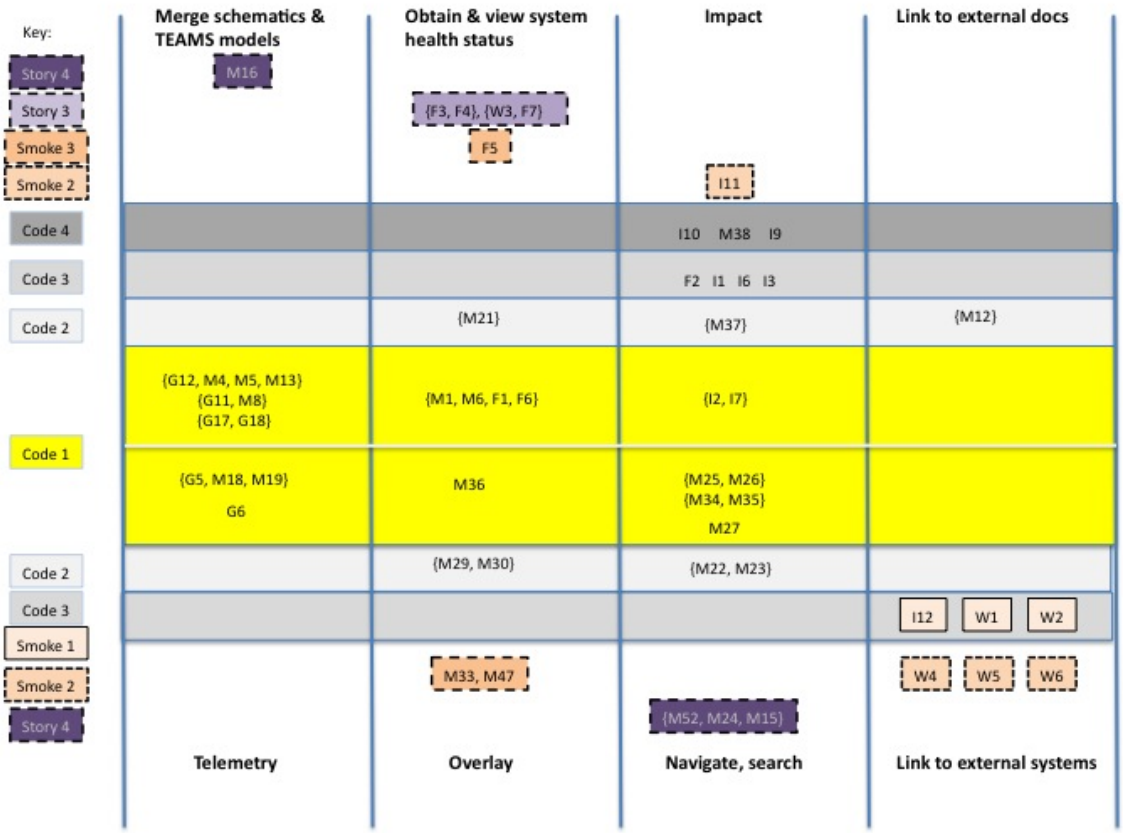


Figure 11: Grouping and prioritization of ACAWS demo capabilities, different view but same information as Figure 10. The eight groups are shown in four columns and two groups with the delineation mark between the upper and lower group shown going between the two yellow rows. The upper groups have the priority 1 items at the bottom whereas the lower groups have them at the top. In this way, we can show the highest priority items together and priority decreases in both (up/down) directions from the middle.

Time limitations precluded implementing all items in the plan specified above. The capabilities showcased in the concept demonstration include the following:

- Reuse of diagrams (SSSH or equivalent), including pan, zoom, jump to "Z"ones
- Reuse of diagnostic models
- Link diagnostic model components to corresponding diagram components
- Ability to use telemetry values
- View telemetry within context of system architecture and distinguish telemetry exceedances
- Ability to display pertinent information in a generally shallow hierarchy.
- Obtain health status of modeled systems within context of system architecture
- Identify if more than one component is suspect
- Ability to view root cause failure for set of failed items (set through telemetry).
- Link external documents to a diagnosis
- Ability to view external documents attached to an item
- View items that would fail for a specified parameter ("MSID")

- Ability to display multiple windows simultaneously

The following section provides details of the concept demonstration implementation.

6.4 User Interface Design Concept

The design concept for the ACAWS concept demo framework, the user interface, and some of the user interaction capabilities is shown in Appendix C.

7. Implementation

The "as built" version of the demo differs from the "as designed" version because some of the capabilities were adjusted due to limitations of the selected development tools. The "as built" version is described in this section.

7.1 Investigative Domain

The foundation of ACAWS is the reuse of diagnostic models, schematics, and other engineering products. For the investigative prototypes, we used QSI's TEAMS product suite and utilized models and engineering products developed for Constellation's Ares I whenever possible. When there was insufficient access to needed products for Ares I, we substituted Shuttle products. In particular, the initial investigations reused the TEAMS models for the Ares I Thrust Vector Control (TVC) system. However, because Ares CAD models are not available to us, we used CAD models from the Shuttle Solid Rocket Booster (SRB) TVC. Ares TVC is closely based on the Shuttle SRB TVC, so we encountered few compatibility issues, none of which were show stopping.[32]

We also used Shuttle standards when Ares standards were not yet available. In particular, ACAWS products were formatted to layout standards available for Shuttle MOD operations products.

7.2 ACAWS Implementation Requirements

The following tables enumerate the implementation-derived requirements and their rationale.

Table 30: Schematic Diagrams.

Req. #	Requirement & *Rationale*
A1.	Obtain schematic diagram drawings in DWG format. *Provides the details of the schematic diagrams in a format that can be translated to SVG format.*
A2.	Translate DWG to SVG format. *Enables access to the primitives (lines, etc.) in XML format; can be translated to custom Java Swing application.*

[32] An example of an issue we ran into regarding Ares I TVC vs. Shuttle CAD models of the Shuttle SRB TVC: the Shuttle CAD drawing with the appropriate area also included the Shuttle main engines as well as the other SRB. Since Ares I has only one SRB, we mapped the left shuttle SRB to the Ares I TVC and ignored the other components for diagnostic modeling. However, we did have some telemetry data for the three main engines and two SRBs.

Table 31: Features of TEAMS that are utilized for ACAWS prototype development.

Req. #	Requirement & *Rationale*
A3.	Access the TEAMS database to retrieve model information. *Enables building the relationship between failure modes and tests for system impact.*
A4.	Access the TEAMS database to retrieve documents and other properties associated with specific TEAMS modules. *Provides one method for attaching documents to system parameters.*
A5.	Use TEAMS-RT to diagnose to component level, with the option to output minimal fault diagnosis. Currently using the TEAMS-RT direct library API rather than the RDS API (which requires running the TEAMS-RDS server). *Presents only the minimal set of components that explain the health status indications (i.e., test results).*

Table 32: Requirements for extensions to TEAMS to enable ACAWS.

Req. #	Requirement & *Rationale*
A6.	Ability to access database information through API. *Direct access to the information through the database (without an API) leaves ACAWS vulnerable to future TEAMS changes of their database structure.*
A7.	Function connectivity not available through API (needed for DFT feedback).
A8.	Command line option to modify models (links, etc.).
A9.	Ability to define propagation (purple) paths as a node list – so that it can used to display ACAWS displays.
A10.	Flip between modes to see how fault propagation changes without having to do a testability analysis after setting the switches each time.

Table 33: Requirements for UI and data system.

Req. #	Requirement & *Rationale*
A11.	Develop user interface in Java Swing. *Enables integration with MCT.*
A12.	Subscribe to ISP for input data source. *Allows for telemetry access, either real-time or from a file playback.*
A13.	Post results to ISP null server. *Allows diagnoses to be used by other MOD programs.*
A14.	Handle data dropouts.
A15.	Handle a large number of data items. For example, the situational awareness vector may contain all measurements and computations (comps).
A16.	Utilize ELOG/Limit management tool for setting annunciation limits for telemetry.
A17.	Utilize MOD sanctioned comps engine for attaching new computations to model.

7.3 ACAWS Implementation Details

Demonstration Scenario – An early step in implementation was to decide on a demonstration scenario that shows the potential of ACAWS in diagnosing a failure. Using an

existing Ares TVC model as a starting point, we initially outlined three hypothetical scenarios as well as two real failures that were logged during earlier shuttle missions. All of the scenarios displayed problems or potential problems in the shuttle SRB TVC (as mentioned previously, a near-exact hardware match to the Ares I TVC). Several metrics went into the choice of scenario, the main ones being listed below:

- Is the failure interesting and/or difficult enough so that ACAWS can add something to the process of fault detection, isolation, and mitigation?
- Are there procedures and flight rules associated with the scenario, in order that we can link to these documents and show added capability?
- Will we be able to get the needed data for such a scenario?
- Are the components involved in the scenario modeled by the current Ares I TVC model?

The chosen scenario is outlined in Section 7.4.

Language/UI Framework - The decision to choose an application framework of Python and Qt was driven by the need to potentially develop for a Linux platform (both Python and Qt are platform independent) as well as the need to build a working prototype in only three months. Python is a very fast, object-oriented development language. Qt is a rich framework with a large user community and provides nearly all the tools needed to build the ACAWS user interface, as well as providing a GUI builder tool that enabled us to create templates of the application prior to adding more complex functionality by hand. Another option for ACAWS GUI development was Java/Swing, but we did not have enough time to implement in such a structured language. The next version of ACAWS has a planned development in Java so that it can be more closely integrated with MCT. Figure 12 shows how the Python/QT interface fits into the ACAWS architecture. In comparison with the high-level architecture shown in Section 6, it should be noted that the *SSSH manager* below is the equivalent of the *Models Manager* while the *Diagnosis manager* is the same as the *Root Cause Manager*. The dashed boxes indicate pieces that are not fully implemented but were incorporated into the design.

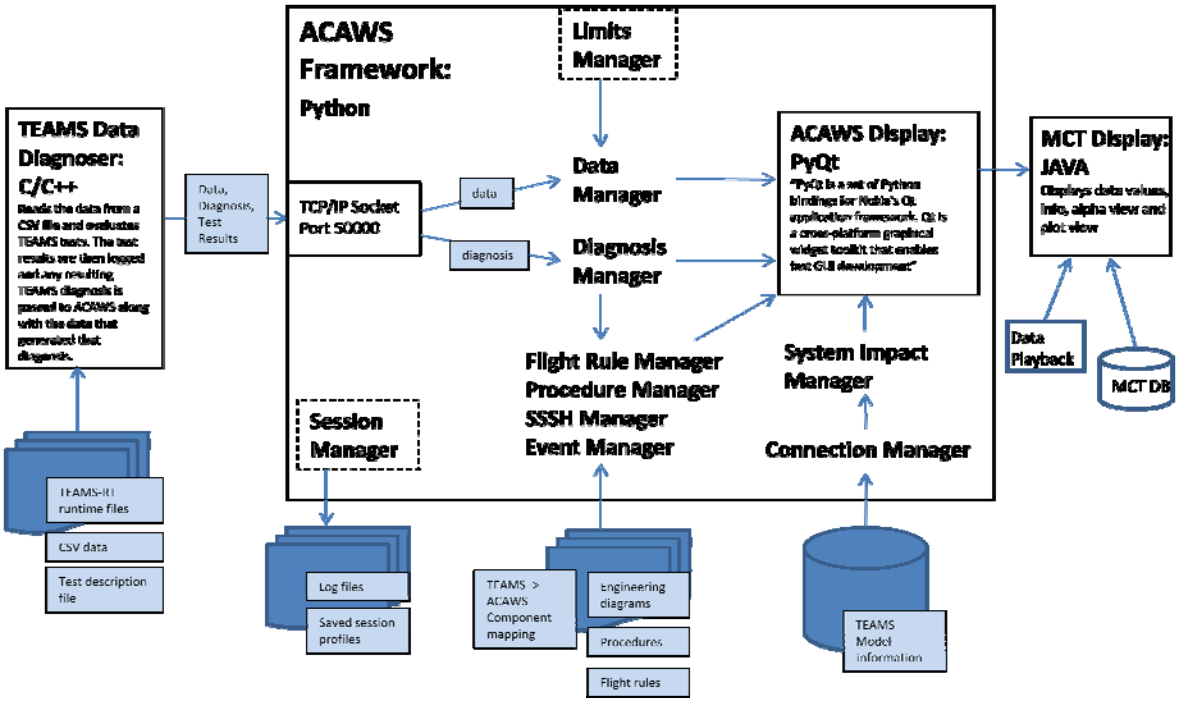


Figure 12: Implementation Architecture

Schematic Representation - The next step was to identify those schematics that were necessary for the development of our demonstration and to convert those schematics into a format that could be used to overlay system health information or model information. As a starting point, we chose three schematics from the Space Shuttle Systems Handbook (SSSH), representing the Main Engine (ME) and Solid Rocket Booster (SRB) actuator physical layout and signal flow, as well as the SRB overview itself. The schematics were converted into a SVG format from the original DWG format for editing and parsing. As we learned more about the tools used to load and view SVG images, we made additional changes to the schematic files in order that they could store useful meta-information about the components represented. An early difficulty faced was how we were to overlay telemetry onto the SSSH schematic – often representing multiple channels and layers of components by overlapping boxes and "zone references" which jump from one place in the schematics to another place in the same schematic or a different schematic. These difficulties, plus the discipline engineer subject matter expert, led us to use a block diagram approach, based on the schematic layout but simplified and expanded to cover other useful telemetry elements that were absent in the original schematic (Figure 13). The multiple layers could then be accessed by clicking on a series of tabs, each representing a Flight Control System (FCS) channel. If the user selected any of the tabs representing FCS channels, ALL of the associated channel tabs changed so as to eliminate the potential confusion of multiple channels being displayed for different components. The channel status information is represented on the left side of the diagram, and the TVC actuators are on the right. The tabs in the center represent the ATVC boxes, in which the automatic channel isolation logic resides.

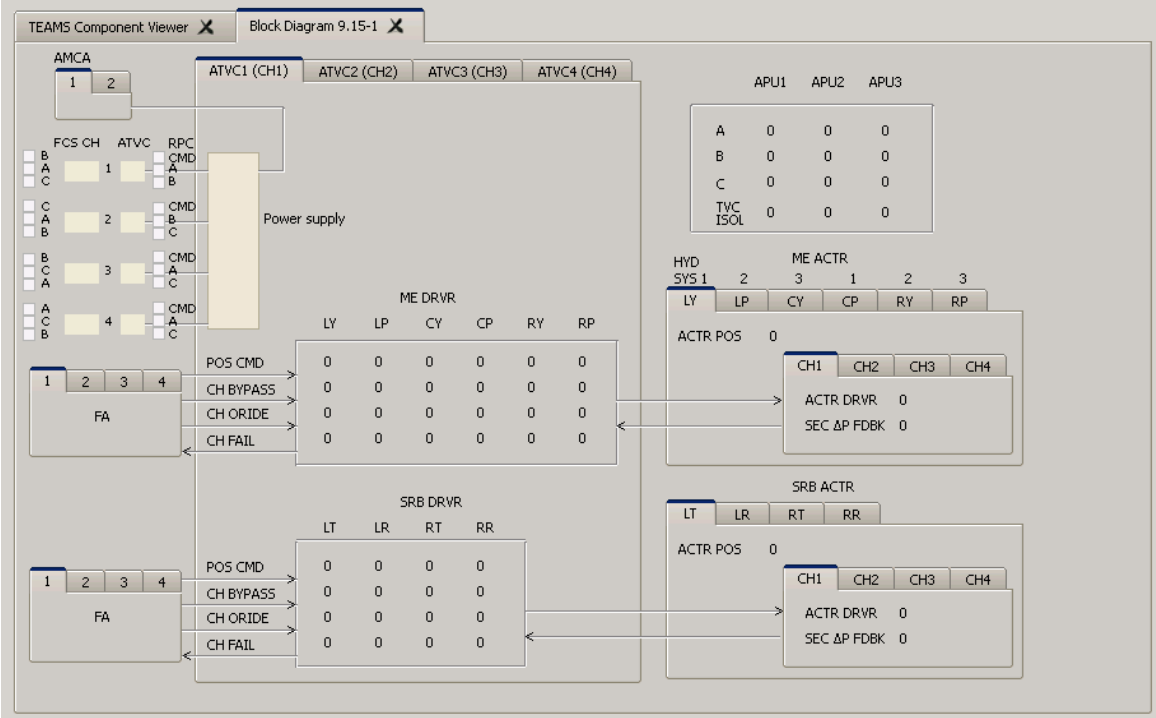


Figure 13: TVC Overview Block Diagram

TEAMS Model Database - Another large piece of the ACAWS software involves integrating the information from the TEAMS model into the underlying system representation. As a first step in this direction, the TEAMS model data first had to be accessed. Unfortunately no formal support for this functionality exists for the tool, so we decided to follow a similar approach as was used for the TEAMS model analyzer software developed at JPL for a peer ISHM project, Functional Fault Analysis (FFA), under development by ARC, MSFC, and JPL. This source code was made available to us, speeding up the development by helping us to understand the schema of the TEAMS database that we needed to query. The TEAMS database is queried and the hierarchy information rebuilt and stored locally along with module properties for the TEAMS model. After the block diagram was completed, the components within TEAMS were mapped to those represented in the diagram by means of an external comma separated values (CSV) file that could be read in at startup. This mapping provided the key to link a component diagnosis from TEAMS to those components in the diagram and SSSH schematics. Figure 14 shows the TEAMS model hierarchy in tree format as read in by ACAWS.

Figure 14: TEAMS Model Hierarchy. (Blurred to hide details.)

TEAMS "Wrapper Code" – After deciding on the TEAMS model to use, the code for evaluating limits, passing or failing TEAMS test results, determining system mode changes and outputting a diagnosis had to be written. Fortunately this wrapper code architecture could mostly be reused from the earlier Ares I-X Ground Diagnostic Prototype (GDP) project. Much of the earlier code was removed to leave only the necessary tests. All prelaunch system and event mode determination could be deleted because our scenario had no relevant events until after liftoff. We also simplified the wrapper code by using only two system modes for the demonstration. Because the TEAMS-RT library is written in ANSI C, the wrapper code also needed to be run from a C/C++ program. The application for reading the scenario data from a CSV file and applying the wrapper code was therefore kept separate from the main ACAWS GUI code. We believe this will more accurately correspond to future integration between multiple user consoles and a centralized diagnosis server running standalone, but it also meant that we needed to develop code to share the diagnosis and test results between applications. We wrote the code to do this using TCP/IP sockets, labeling the type of information passed by using XML tags and ordering or prioritizing the data on the ACAWS side.

Capturing the Data - After deciding on a scenario, the data needed to be generated and saved in some format so that we could replay it as necessary. The data file generation took place using the Flight Controller Trainer (FCT) at JSC, in which failure scenarios can be injected for training and testing. It is possible to specify a list of MSID values to be stored and exported while the trainer is running. The original file containing these values was provided to us in SITF format, which needed to be converted to CSV in order that the

TEAMS wrapper code could read through it and evaluate the pass/fail results for a diagnosis. There were also some minor changes made to the converted CSV file, such as converting timestamps to a readable format and adding one additional MSID to represent the channel fail flag during the first failure. Because all of the MSID values were provided at the same rate (1Hz), some of the more complicated timing issues were avoided when dealing with multiple-rate data.

Playback of Data - The data is served by playing back a CSV data file. The C/C++ program is written to read the data file and send it to the ACAWS GUI. The CSV file contains GMT time, mission elapsed time (MET) and a vector of data values at each row. Software will play back the data according to the time interval between each row. Current data rate is approximately 1Hz. Since the front end of the ACAWS GUI is written in Python, the playback data must be sent to the GUI using a language neutral method. We use a TCP/IP socket connection to pass data in this implementation.

TEAMS Model – Because the existing Ares I TVC model was developed to support only pre-launch operations, it did not include (represent) certain components needed for the demo scenario's (post-launch) fault. Some model additions had to be made, specifically adding the upstream command and control signals that were being sent from the Flight Aft (FA) Multiplexer/Demultiplexer (MDM) and the ATVC. These signals are associated with the secondary delta pressure tests at the actuator and provide the first diagnosis signature during the demo scenario. We also added a test to be triggered for the FA1 BCE Bypass flag in order to correctly implicate the FA1 MDM after the first failure.

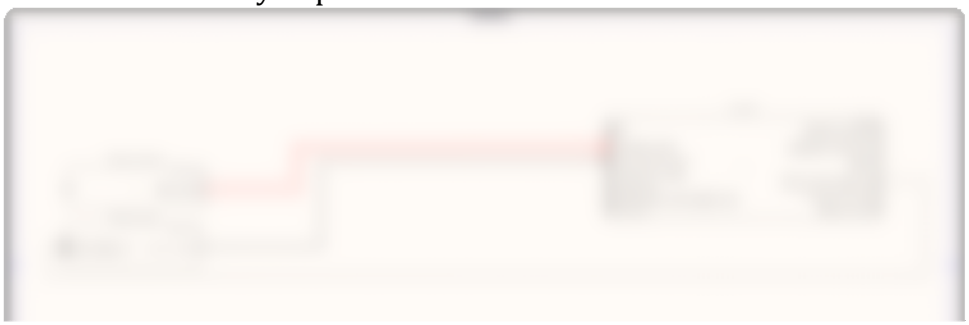


Figure 15: TEAMS Model, top-level view (blurred on purpose to hide details)

The high-level organization is shown in Figure 16. The top-level model (shown both in Figure 15 and in the upper left corner of Figure 16) has both Avionics and TVC hardware subsystems. These subsystems in turn have modules for avionics hardware (GPC, MDM, ATVC) and TVC hardware (FSM, APU, HPU and Actuator). The TEAMS Designer model-building tool supports the hierarchical and distributed modeling of failure modes and test points.

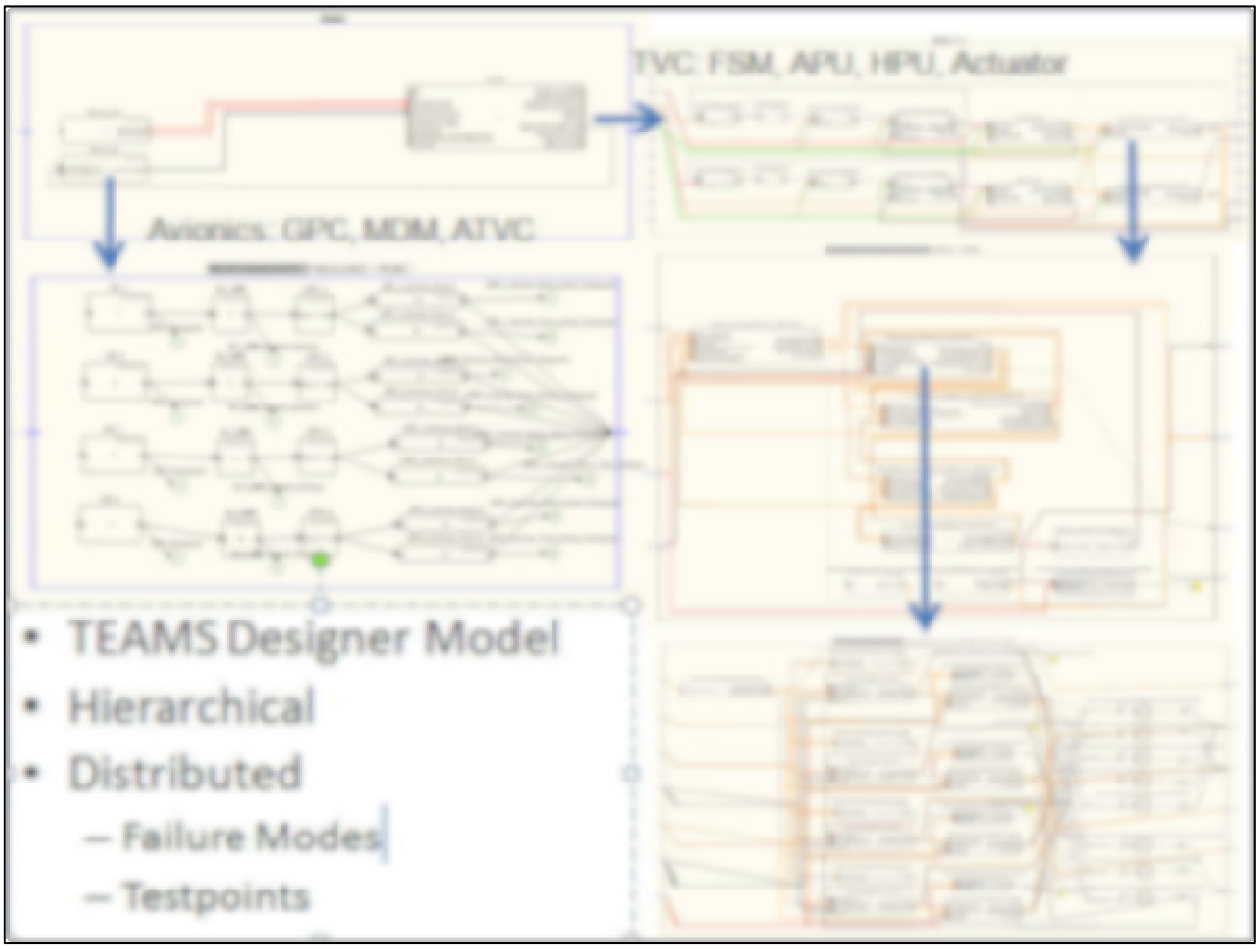


Figure 16: TEAMS Designer Model is constructed from Avionics and TVC models. The model is hierarchical with distributed failure modes and test points. The full model is compiled into a D-matrix (one per system mode).

TEAMS D-matrix – The TEAMS Designer model is converted into a D-matrix to support real-time diagnosis, as shown in Figure 17. Whereas a path between a failure mode and a test point can be followed in the TEAMS Designer model by following links, in the D-matrix – where the failure modes are rows, the tests are columns – if a path exists between a failure mode and a test, there is a one (1) for the matrix element for that failure mode and test pair. That is, the D-matrix represents the existence of a path between a failure mode (rows of the matrix) and a test (columns of the matrix) by inserting a "1" (i.e., path exists) for that matrix element. If there is no link between a test and a failure mode, the corresponding matrix element is set to "0" (i.e., path does not exist; the "0" is usually implied and not explicitly filled in).

The relevant portion of the TEAMS D-matrix for ascent phase is shown in Figure 18. The columns of the D-matrix define the twenty-six (26) tests. The rows of the D-matrix define the fifty-two (52) failure sources. Again, where a relationship exists between a failure mode and a test there will be a one (1; shown in red) in the matrix element.

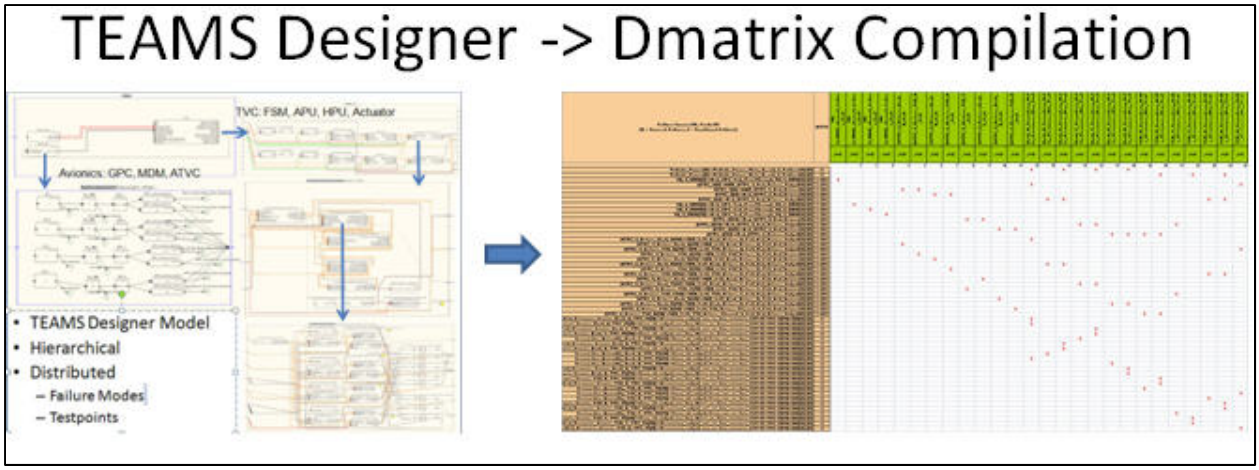


Figure 17: TEAMS Designer Model Converted to D-matrix.

For a given failure source row, all the "1"s in the row define the tests which can detect the failure source. For example given:

- (row) failure source: ATVC_4[12]->Bad_ATVC_4_Driver_Output
- (set of columns) tests which can detect it are defined (all the 1s in the row):
    - Rock_D_Secondary_Delta_Pressure_Min_Test
    - Rock_D_Secondary_Delta_Pressure_Max_Test
    - Tilt_D_Secondary_Delta_Pressure_Min_Test
    - Tilt_D_Secondary_Delta_Pressure_Max_Test

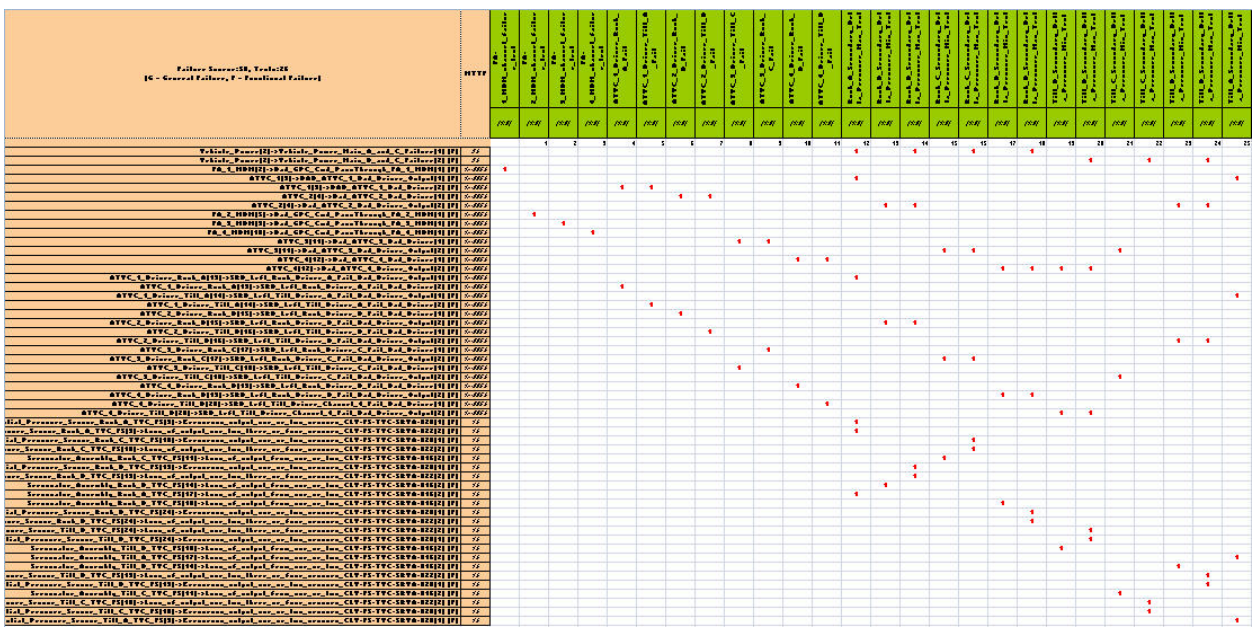


Figure 18: TEAMS D-Matrix for Ascent Phase.

For a given test column, all the "1"s in the column define the ambiguity group for that test, i.e. all the failure sources which the test can detect. For example, given:

- (column) test: Rock_A_Secondary_Delta_Pressure_Max_Test
- (set of rows) failure modes which match detection signature (all the 1's in the column)
    - ATVC_1[3]->BAD_ATVC_1_Bad_Driver_Output[1] (F)

- ATVC_1_Driver_Rock_A[13]->SRB_Left_Rock_Driver_A_Fail_Bad_Driver_Output[1] (F)
- Servovalve_Differential_Pressure_Sensor_Rock_A_TVC_FS[9]->Erroneous_output_one_or_two_sensors_CLV-FS-TVC-SRVA-020[1] (F)
- Servovalve_Differential_Pressure_Sensor_Rock_A_TVC_FS[9]->Loss_of_output_one_two_three_or_four_sensors_CLV-FS-TVC-SRVA-022[2] (F)
- Servovalve_Assembly_Rock_A_TVC_FS[17]->Loss_of_output_from_one_or_two_CLV-FS-TVC-SRVA-016[2] (F)

At runtime, the tests in the D-matrix are evaluated by the "wrapper code" and result in values of pass or fail. The specification of the tests, shown in Figure 19, uses a Test Description Format (TDF) file developed for the Ares I-X GDP project. To quote the GDP project final report, "The TDF file provides the parameters required by the test logic for each test defined in the TEAMS model. The TDF is read by the wrapper code during initialization to allocate the memory required for the tests, and once the wrapper code begins executing, it only needs to pass the updated MSID values to the test logic functions to generate the pass and fail results that will be passed to TEAMS-RT." The TDF specification maps telemetry information to the TEAMS model and can handle three types of test logic: consistency check, to verify that a two-position remotely controlled component with feedback (e.g., a valve) is in the position to which it was commanded; transducer tests, which compare an analog value to a threshold to determine whether the measurement is out of its expected range (both high and low limits); and discrete tests, which verify that a component with a discrete output is in the correct state for a given condition. In deciphering the specification file shown in Figure 19, note that the first character specifies the test type, with "d" representing discrete tests and "t" representing transducer (i.e., analog) tests. A detailed description of TDF is available in the Ares I-X GDP project final report.[33]

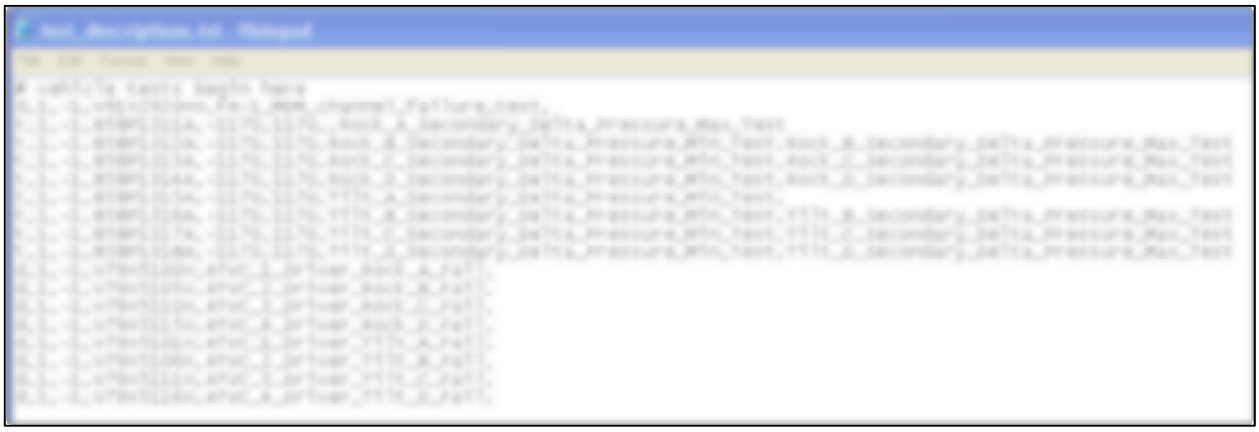


Figure 19: TEAMS RT Wrapper Specification in Test Description Format (TDF). (Blurred to hide detail.)

User Interface Design – The interface was designed to be extremely customizable for the operator, so that he or she could rearrange or hide/show the location of tools on the desktop as needed. Qt provides a method to do this through dockable windows, in which a

[33] For the Ares I-X GDP project final report, contact Mark.A.Schwabacher@nasa.gov.

main center frame is surrounded by docking windows that can pop-out and become independent windows or can be rearranged in any order around the central window. If one dockable window is hidden or minimized, the windows around it automatically resize to use that space. Windows can also be overlaid on top of one another as part of a tabbed pane. Figure 20 shows an example of the default layout, with each specific window described below.

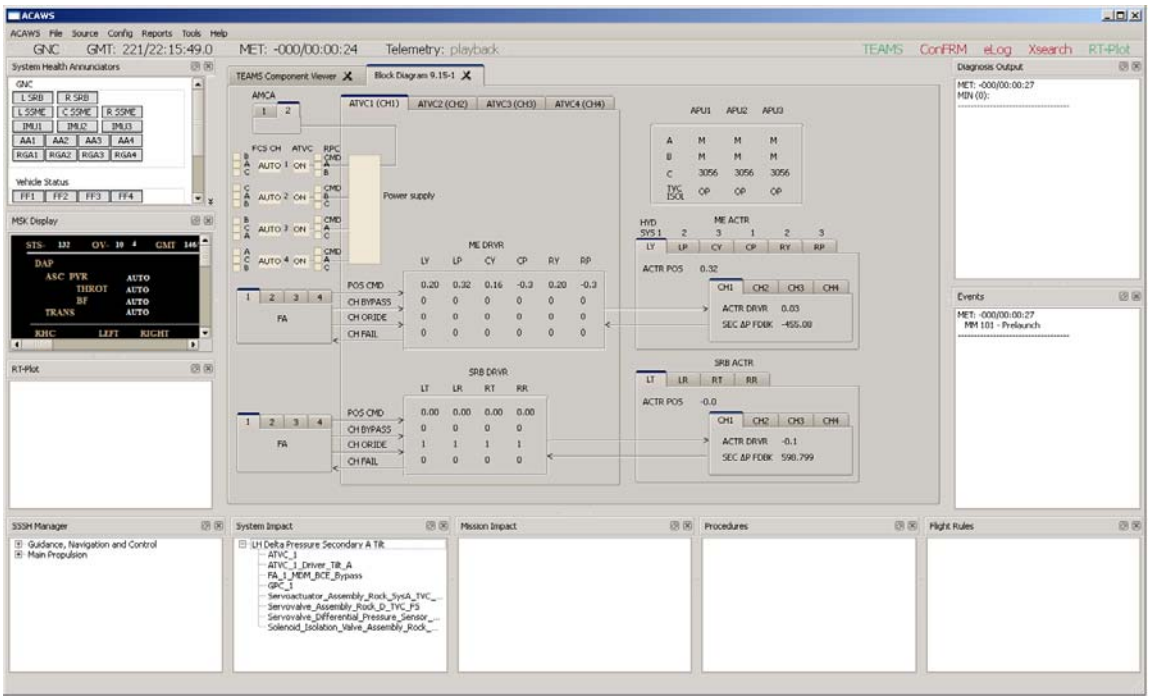


Figure 20: ACAWS Default Layout.

Header – The header bar of the display shows the Discipline designator, the Greenwich Mean Time (GMT) and Mission-Elapsed Time (MET), the Telemetry source (playback, live, etc.), and the status of the tools that are currently connected/associated with ACAWS.

System Health Annunciator - The health annunciator can be customized by a user to include the systems or components that they would like a health status for. The health annunciator panel is broken up into two areas: a discipline specific area and vehicle status area. The discipline-specific annunciator panel can be constructed to contain information specific to a discipline. The vehicle status annunciator can be constructed to include systems information considered outside of a specific discipline. The *System Health Annunciator* can show varying hierarchy and can include details down to the component level or as high as the system level.

MCT Tabular Display (represented by MSK display shown) - All systems may not lend themselves to a functional systems layout. In that event, the *MCT Tabular Display* window allows for viewing of MCT-built tabular displays.

RT-Plot - The *RT-Plot* window displays plotting/trending information of an MSID(s) that are obtained through MCT. This functionality was not fully integrated for the demonstration but instead opened in a separate MCT window for viewing.

SSSH Manager - The *SSSH Manager* allows for viewing of/interaction with SSSH (or equivalent) drawings. Interaction includes zoom in/out, identification of failures on schematic, and selection of the 'Z'one symbol for schematic to schematic interconnectivity just to name a few.

Diagnosis Output - The TEAMS model diagnosis output is displayed in the *Diagnosis Output* window. Each diagnosis item is displayed along with the timestamp at the time it was diagnosed.

Events - The *Events* window contains enhanced ELOG-like function that captures events (Vehicle Major Mode identification, failure description, etc). Unlike ELOG, when a higher system level fault occurs, 'Events' shows the message associated with the failure and not 20 other possible C/W messages that could be associated with the fault.

System Impact - The *System Impact* window provides traceability between a selected MSID and failure modes that are tied to that MSID.

Mission Impact - In the future, the *Mission Impact* window will identify how a fault impacts the mission timeline.

Procedures - The *Procedures* window allows for the displaying of procedures. Once a diagnosis is selected by double clicking on it, the procedure(s) associated with the fault appear as a tab behind the 'Procedures' window.

Flight Rules - The *Flight Rules* window allows for the displaying of Flight Rules. Once a diagnosis is selected, the Flight Rule(s) associated with the fault appear in this window.

TEAMS Component Viewer - Provides hierarchy of TEAMS model without pulling up schematics.

Block Diagram 9.15-1 - The block diagram is a functional depiction of a schematic (SSSH 9.15-1) with over-laid telemetry and state information included.

The layout of the block diagram is more in line with how the system is learned during training. It provides the user with color-coding to draw attention to a parameter when a pre-defined limit has been exceeded (yellow or red) or the data is missing (no longer being received; shown in magenta).

Hovering over a specific piece of telemetry provides the user with the associated MSID. Right clicking a telemetry term provides a menu with options, including the MCT view (RT-Plot, alpha numeric, etc) of the MSID.

In Figure 21, the same interface is shown after a failure has been injected to display some of the added functionality mentioned above.

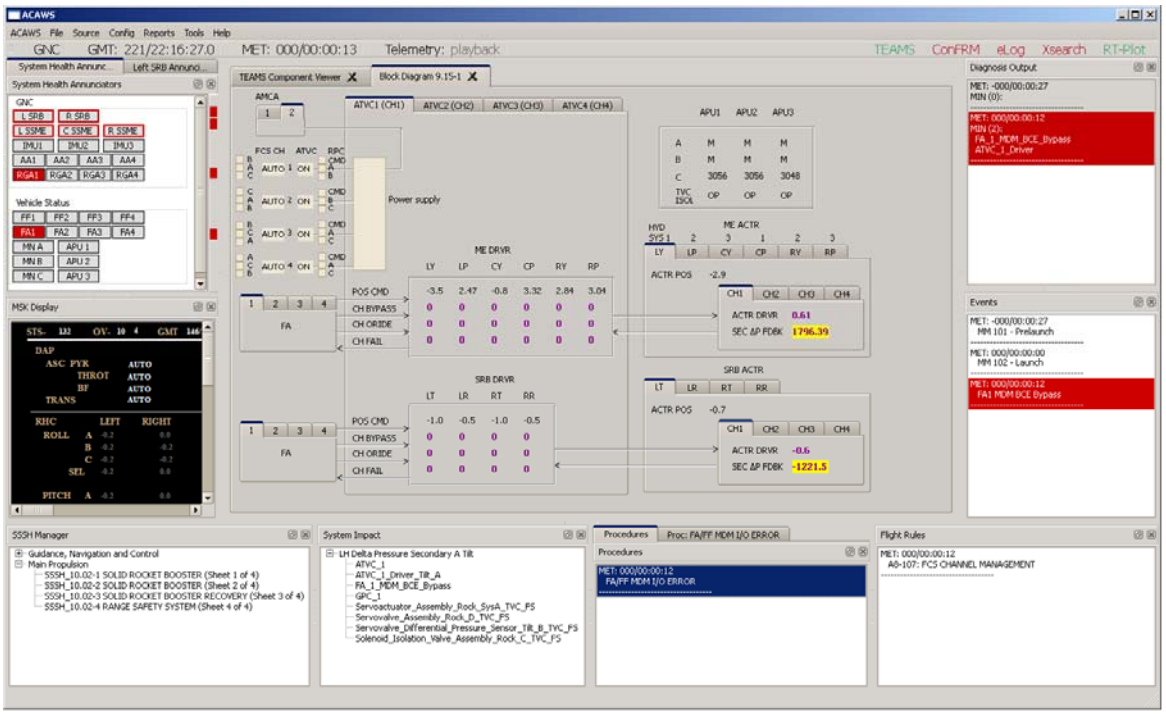


Figure 21: Post-diagnosis ACAWS Display

Approval by System Experts – After developing a working demonstration, much of the effort went into adding user interface features that are useful for the flight controllers and into changing features to match controller expectations. We worked with a Guidance, Navigation, and Control (GNC) flight controller regularly over the last month of the design. The user observations during this period of development were useful in understanding where current operator practices should be reused in UI design. Some of the lessons learned during this phase of development are enumerated in Section 8.4.

Integration with MCT – ACAWS was loosely integrated with MCT using basic Windows OS system calls. MCT provides functionality that ACAWS would otherwise have to integrate with separately, such as an RT-Plot trend display or MSK tabular display. When requested, ACAWS can invoke MCT windows from its GUI interface. When the operator needs to see more detailed information about an MSID, he or she can right click on a data value in the block diagram and bring up the MCT window to display the multiple different views that MCT provides (*Alpha* view, *Plot* view or *Info* view). This level of integration is a very simple way to show both tools working together; it does not leverage maximum capabilities of MCT software. We took this approach only because of development time constraints of this prototype requirement. To leverage maximum capabilities of MCT software, the ACAWS GUI must be implemented in Java. Implementing ACAWS within the MCT framework as a plug-in component will better utilize MCT capabilities.

7.4 Demonstration Scenario

The demonstration scenario that we chose reflects a shuttle ascent during which time the ACAWS user is monitoring the SRB TVC and Main Engine (ME) TVC (see Table 34 for the complete timeline of events for the demonstration scenario). The data file used is approximately three minutes in duration and starts at 27 seconds prior to launch (Mission Elapsed Time (MET) T-27 seconds; Step 1, see Table 34). At MET T-17 seconds before launch, the TVC is gimbaled (Step 2, see Table 34).

Table 34: Demonstration scenario timeline of events.

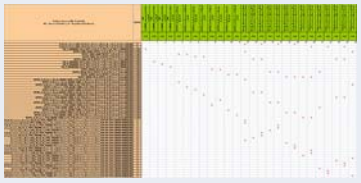
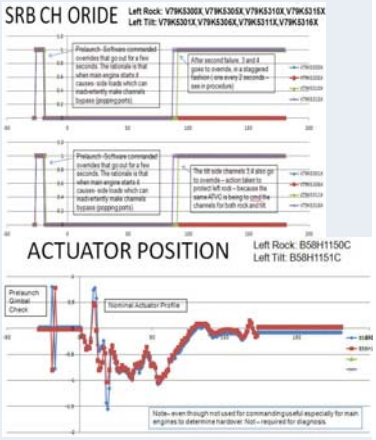
Step	MET	Event	TEAMS Diagnosis	MSID
1	-27.7596	Start of simulation	<none>	
2	-17.8812	Start: Prelaunch SRB Gimbal Check	<none>	
3	-15.8796	End: Prelaunch Gimbal Check	<none>	
4		Launch	<none>	
5	12.6	First Failure Installed – FA 1 MDM BCE-bypass = 1	<FA1 MDM Failure>	Bus Error – loss of signal
6		Out of family driver current for CH1	<FA1 MDM Failure>	How to really tell that it is out of family,
7	17.4816	high Sec DP (Left Rock Ch1 B58P1311A);	<FA1 MDM Failure>	Never reaches, MCC informs crew to take channel off line? (in their procedure)
8	21.4416	FCS CH1 Off	<FA1 MDM Failure>	Crew takes channel of-line via D&C;
9	57.9204	Second Failure Installed	<FA1 MDM Failure>	
10	57.9204	HighSec DP (Left Rock Ch 2 **B58P1312A)**	<FA1 MDM Failure> (actual) <sensor failure> <sensor excitation power> <bad ATVC servovalve driver circuitry> (actual) <bad ATVC isolation driver circuitry> <channel failure on actuator> <servovalve failure>	Confirming cues to rule out ambiguity group elements – <bad MDM>, <bad GPC>, <bad ATVC> Low Prob: <servovalve failure> <SRB MDMS, IEAs> -
11	59.91	FCS CH2 Fail	<FA1 MDM Failure>	

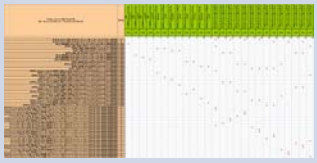
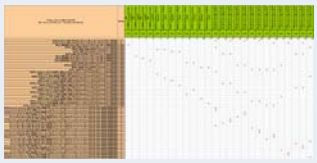
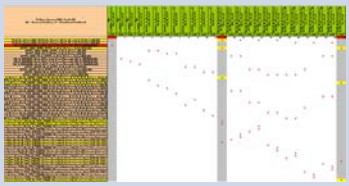
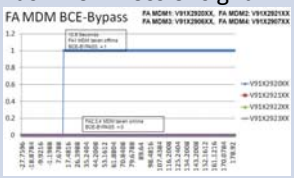
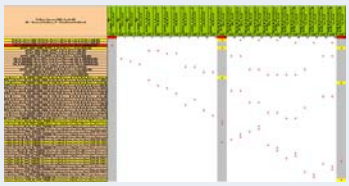
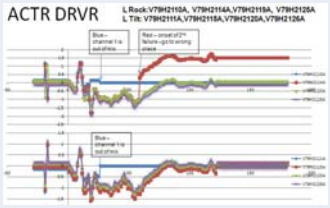
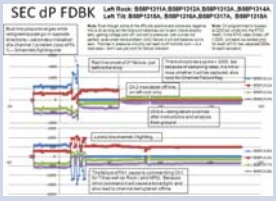
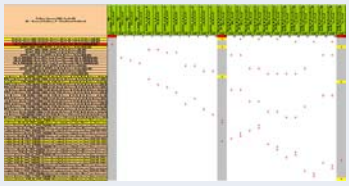
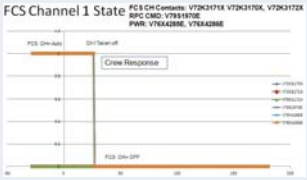
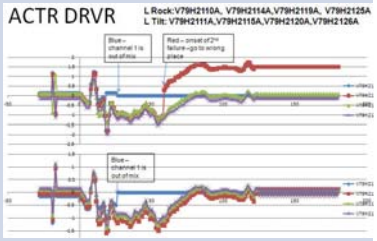
		(V79X5105X)	<bad ATVC servovalve driver circuitry>	
12	87.6	Crew Takes Channels 3,4 to Override	<FA1 MDM Failure>	Follow procedure
13	123.1596	Start of SRB Separation	<FA1 MDM Failure>	
14	179.9604	End of simulation		

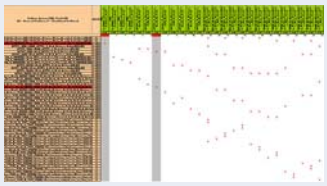
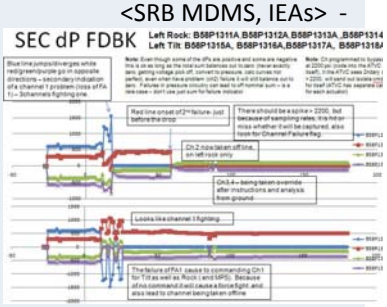
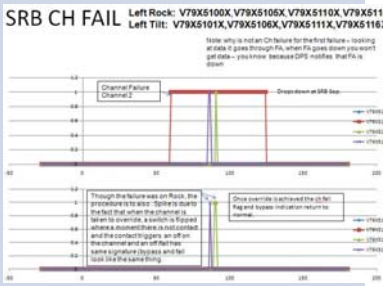
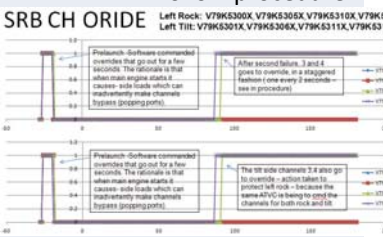
Shortly after liftoff (MET T+12.6 seconds), FA1 MDM fails and as a result, FCS Channel 1 is turned *OFF* by the crew. At MET T+57.6 seconds, a second failure, ATVC 2 Rock Driver B, fails due to a high secondary delta pressure measurement and is automatically bypassed. This drives the crew to take action by taking FCS Channels 3 and 4 to the *Override* position in order to avoid the situation of the remaining good channels from being bypassed. The correlation of TEAMS RT D-matrix diagnosis and the MSID strip charts is presented in Table 35. After the table, the steps in the timeline are summarized into the following categories:

- Steps 1 – 4: Prelaunch Gimbal, MPS Start
- Steps 5 – 7: First Failure
- Step 8: First Failure Recovery
- Steps 9 – 11: Second Failure
- Steps 12: Second Failure Recovery
- Steps 13-14: SRB Sep / End of Simulation

Table 35: Demonstration scenario timeline of events with TEAMS and MSID strip chart annotations.

Step	MET	Event	TEAMS Diagnosis	MSID
1	-27.7596	Start of simulation	<none>	
2	-17.8812	Start: Prelaunch SRB Gimbal Check	<none> 	

3	-15.8796	End: Prelaunch Gimbal Check	<none> 	
4		Launch	<none> 	
5	12.6	First Failure Installed – FA 1 MDM BCE-bypass = 1	<FA1 MDM Failure> 	Bus Error – loss of signal 
6		Out of family driver current for CH1	<FA1 MDM Failure> 	
7	17.4816	high Sec DP (Left Rock Ch1 B58P1311A);	<FA1 MDM Failure>	
8	21.4416	FCS CH1 Off	<FA1 MDM Failure> 	Crew takes channel of-line via D&C; 
9	57.9204	Second Failure Installed	<FA1 MDM Failure>	

10	57.9204	High Sec DP (Left Rock Ch 2 **B58P1312A)**	<FA1 MDM Failure> (actual) <sensor failure> <sensor excitation power> <bad ATVC servovalve driver circuitry> (actual) <bad ATVC isolation driver circuitry> <channel failure on actuator> <servovalve failure> 	Confirming cues to rule out ambiguity group elements – <bad MDM>, <bad GPC>, <bad ATVC> Low Prob: <servovalve failure> <SRB MDMS, IEAs> - 
11	59.91	FCS CH2 Fail (V79X5105X)	<FA1 MDM Failure> <bad ATVC servovalve driver circuitry> 	
12	87.6	Crew Takes Channels 3,4 to Override	<FA1 MDM Failure> 	Follow procedure 
13	123.1596	Start of SRB Separation	<FA1 MDM Failure> 	
14	179.9604	End of simulation		

Steps 1 – 4: Prelaunch Gimbal

Initially the simulator is started, the prelaunch gimbal checks occur, main engines are started, and the channels are taken to override temporarily in order to prevent "popping ports" when the main engines start.

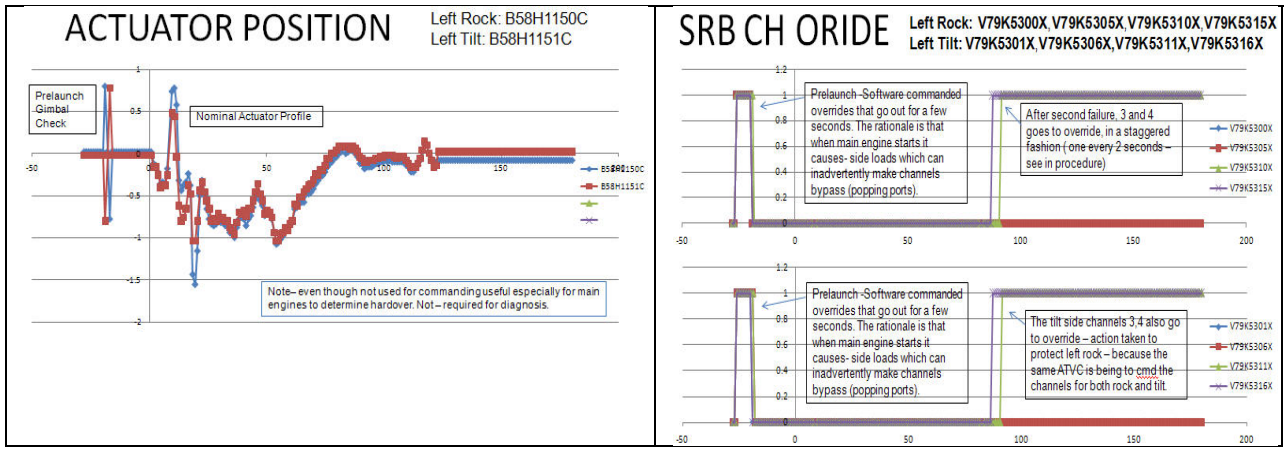


Figure 22: Step 1 – 4: Prelaunch Relevant MSIDs. Relevant portions of Actuator Position (prelaunch), relevant portions of SRB CH ORIDE – prelaunch override.

No diagnosis – all is working at this stage as evidenced by the no active test columns and fault mode rows.

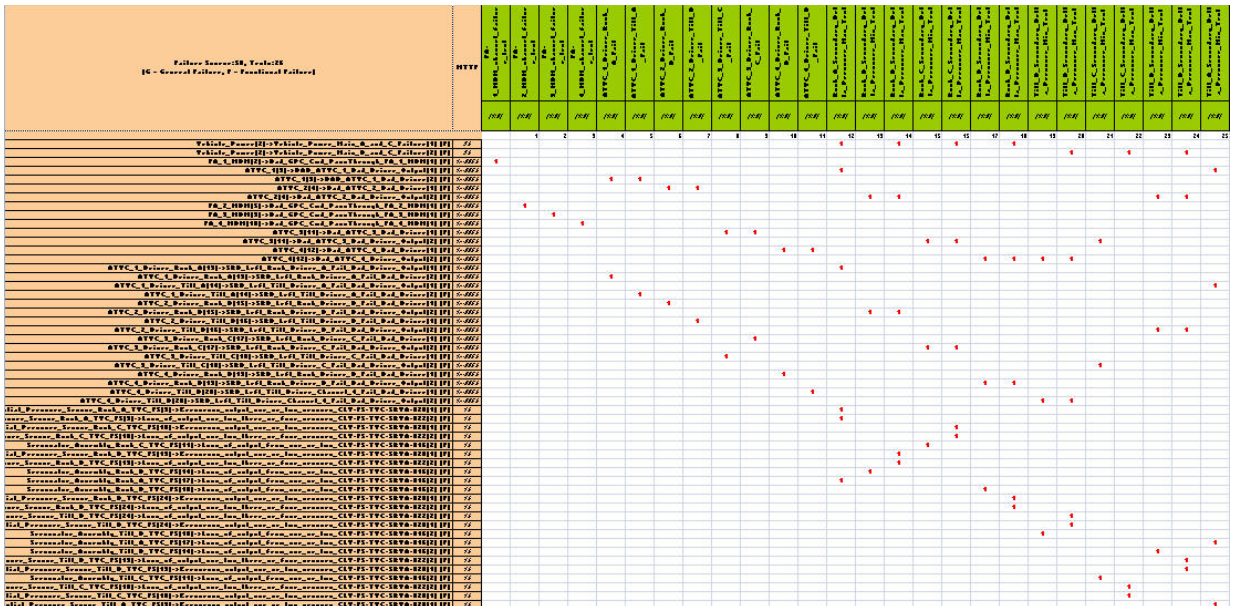


Figure 23: Steps 1 – 4: TEAMS Diagnosis – No Faults.

Steps 5-7: First Failure

The first failure occurs when the FA1 MDM is no longer in contact (FA MDM BCE-Bypass). This implies that the ATVC-1 will no longer receive DAP command updates from GPS through MDM to ATVC. Hence the ATVC continues sending out the last value (or maybe a zero?) (ACTR DRVR). However, since ATVC-2,3,4 are still receiving their new commands (ACTR DRVR) which will conflict with this constant signal from CH 1. The conflict is realized when the servo valves work against each other and the secondary delta pressure of the channel (SEC dp FDBK) rise to redline limits.

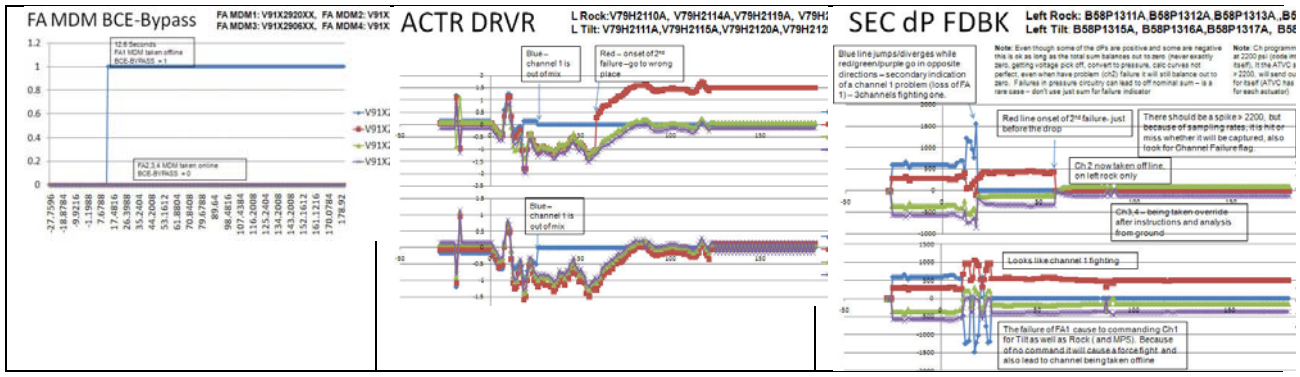


Figure 24: Step 5 – 7: First Failure MSIDs. When FA-1 MDM BCE-Bypass Flag goes high, then the control path to the actuator is cut. Actuator Driver current gets stuck at constant and servo valve delta-pressure sensor starts to increase as force-fight with remaining three channels ensues.

The diagnosis is based upon tests that used the MSIDS for FA MCMBCE-Bypass, ACTR DRVR and SEC dP FDBK. In TEAMS RT, twenty tests pass[34] and three tests fail[35], specifically:

- FA-1_MDM_channel_failure_test
- Rock_A_Secondary_Delta_Pressure_Max_Test,
- Tilt_A_Secondary_Delta_Pressure_Min_Test

As highlighted in Figure 25, these three tests (highlighting the three grey columns in the D-matrix) determine a single diagnosis:

- Bad_GPC_Cmd_PassThrough_FA_1_MDM[1]<-FA_1_MDM[2]<-SRB_ATVC_Command_Flow[21]<-Vehicle_Cmd[3] (see red row)

and a set of suspects (highlighted rows in yellow).

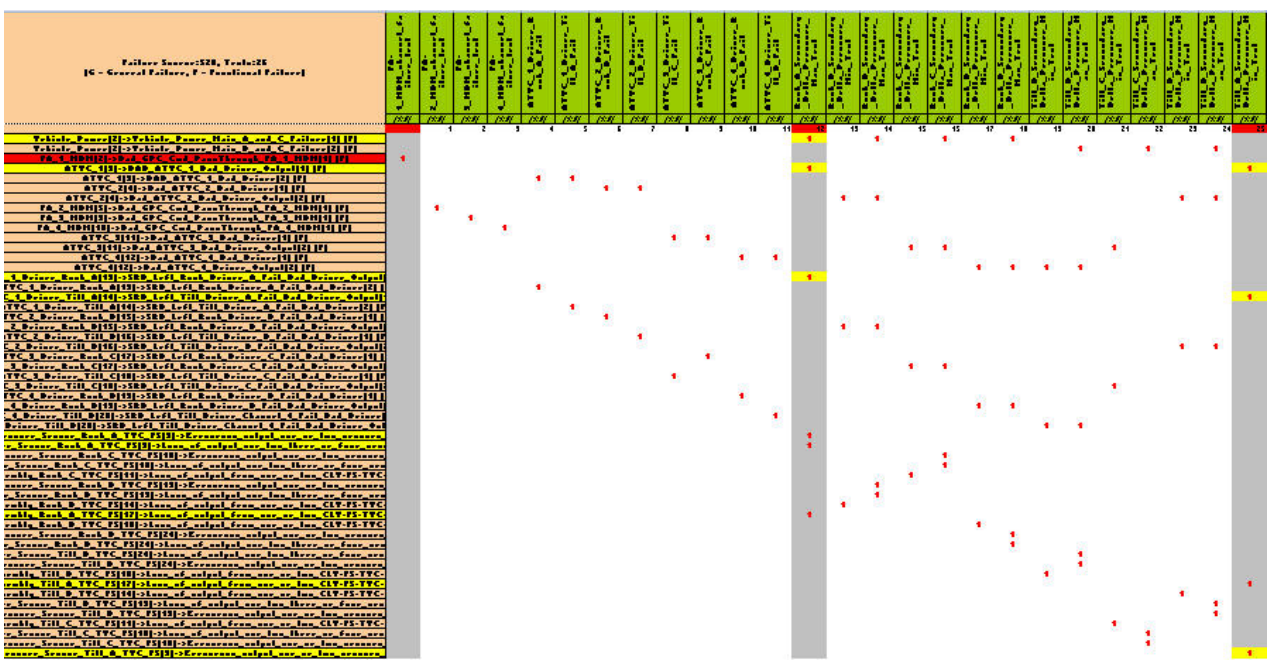


Figure 25: Steps 5 -7 GMT: 221:22:16:26.0; MET: 13.6 TEAMS Diagnosis First Failure- FA-1 MDM Failed.

[34] Steps 5 – 7 passed tests: 4 5 6 7 8 9 10 11 13 14 15 16 17 18 19 20 21 22 23 24

[35] Steps 5 – 7 failed tests: 0 12 25

Step 8: First Failure Recovery

Following the Failure, Impact, Workaround (FIW) approach to fault management, once the failure is isolated the crew can affect the workaround and the crew can take the errant channel offline (D&C controls), as shown in Figure 26. The TEAMS diagnosis continues to be a failed FA-1 MDM.

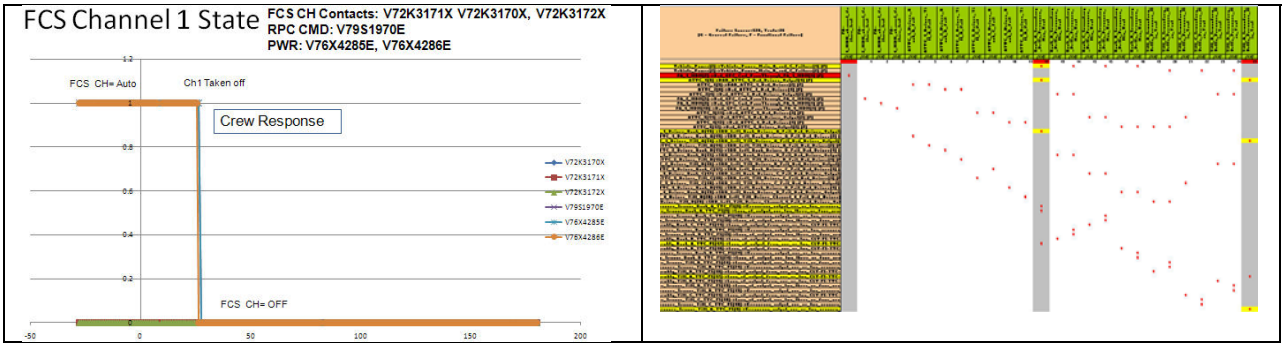


Figure 26: Crew recovery from first failure: take Channel 1 Offline.

Steps 9 – 11: Second Failure

Continuing on during ascent, the second failure occurs when an actuator driver current goes out of family and stays hard over (ACTR DRVR). This causes a force-fight on the channel 2, seen in SEC dP FDBK in Figure 27.

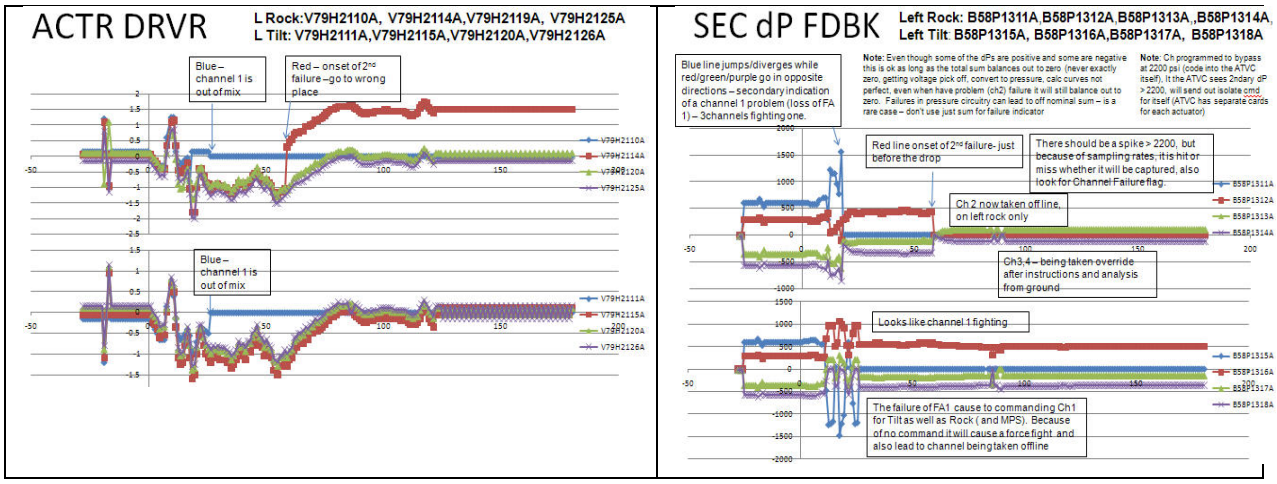


Figure 27: Steps 9 – 11: Second Failure MSIDs.

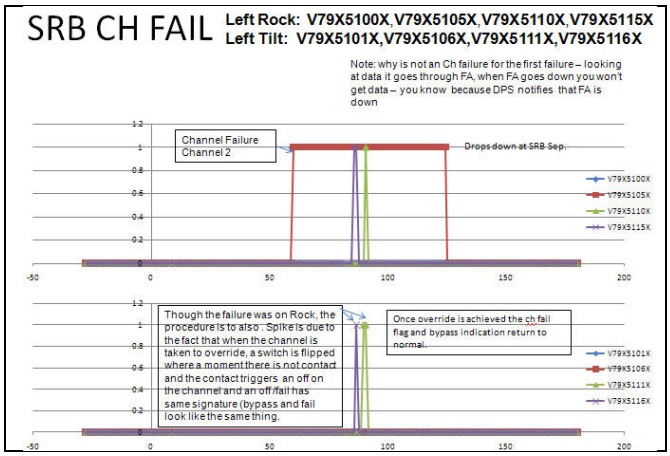


Figure 28: Steps 9 – 11.

Once the secondary delta pressure (SEC dP FDBK) reaches more than 2200 psia, the solenoid isolation valve is activated and the channel is failed. (SRB CH FAIL, shown in Figure 28).

The diagnosis now is a double fault, including the first fault that continues to linger (loss of FA-1 MDM) and now adding the failure of the ATVC Driver for channel 2 (SRB CH FAIL). The diagnosis is based upon tests that used the MSIDS for SEC dP FDBK and SRB CH FAIL. The TEAMS RT, twenty-one tests passed[36] and two tests failed[37], specifically the following:

- FA-1_MDM_channel_failure_test
- ATVC_2_Driver_Rock_B_Fail

As highlighted in Figure 29, these two failed tests (grey columns in the D-matrix) determine the multiple fault diagnoses (two red rows):

- Bad_GPC_Cmd_PassThrough_FA_1_MDM[1]<-FA_1_MDM[2]<-SRB_ATVC_Command_Flow[21]<-Vehicle_Cmd[3]
- SRB_Left_Rock_Driver_B_Fail_Bad_Driver[1]<-ATVC_2_Driver_Rock_B[15]<-SRB_ATVC_Command_Flow[21]<-Vehicle_Cmd[3]

[36] Steps 9 – 11 passed tests: 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25.
[37] Steps 9 – 11 failed tests: 0 6.

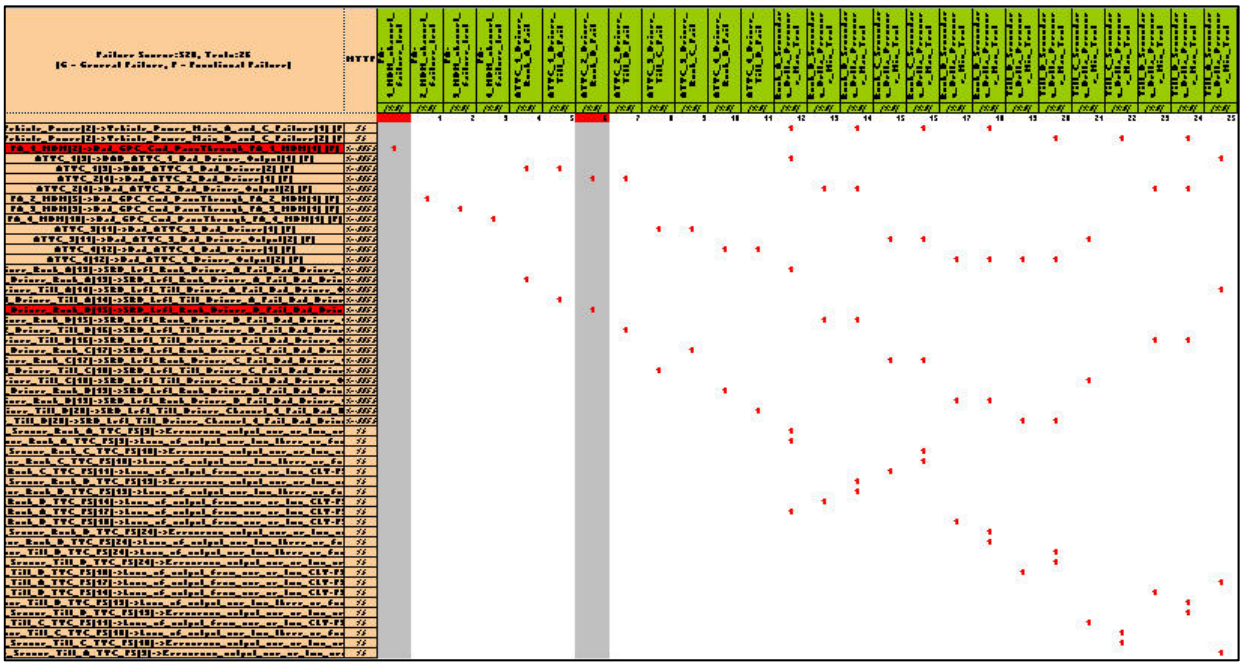


Figure 29: Steps 9 -11 221:22:17:14.0; MET Seconds: 59.9204 TEAMS Diagnosis First Faillure + Second Failure: FA-1 MDM Failed, ATVC_Driver_Ch2_Failed

Step 12: Second Failure Recovery

To counteract the failure of the second channel and to follow procedure (more FIW), the crew take channels 3 and 4 to override – to prevent isolation valve logic from removing any of the two remaining inputs to the powerspool of the servo-mechanical actuator.

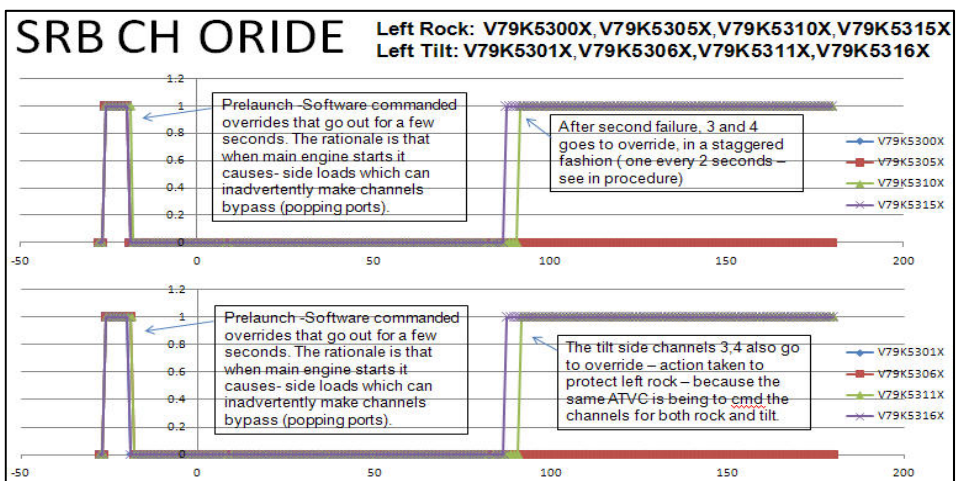


Figure 30: Crew recovery from second failure: take Channels 3 and 4 to override.

Step 13 – 14: SRB Sep / End of Simulation

At the end of the data, the ATVC failure is gone – stands to reason since the SRB has been jettisoned and the orbiter with ATVC is gone.

8. Lessons Learned

8.1 Schematic Diagrams

The SVG (Scalable Vector Graphics) format works well as a starting point for ACAWS. SVG format is an open standard family of specifications of an XML-based file format for describing static and dynamic (animated) 2-D vector graphics. Because SVG images and behavior are defined in XML text files, they can be searched, edited with any text editor, and rendered by most web browsers. Converting the schematic drawings into SVG format allows for their translation into our own Java Swing application. Support for SVG format is well established and parsers are available in most programming languages.

As mentioned earlier, we used schematics from the SSSH schematic handbook. Shuttle schematic drawings are currently available from JSC/Bastion in the following electronic formats:

- DNG, "digital negative", format for raw files generated by digital cameras,
- DWG, "drawing", format for storing 2D and 3D design data and metadata; native format for several CAD packages including AutoCAD,
- CGM, "computer graphics metafile", format for 2D vector graphics; free and open international standard, and
- DXF, "data exchange format", CAD data file format developed by Autodesk for enabling interoperability between AutoCAD and other programs.

Converting DNG to SVG results in a loss of many details so we quickly abandoned this path. CGM and DXF formats available from Bastion also showed a large loss of details (the integrity of the files was negatively affected). This may be a result of how they generated the CGM and DXF format files (possibly by converting from DNG or DWG) or because the formats are not suitable for representing schematic diagram information. We did not pursue this path after seeing the converted drawings. Converting DWG to SVG works well and provides access to and location of each of the primitives on the schematic diagram.

The amount of effort needed to bring schematic diagrams up to the point of providing useful information about the structure of the underlying vehicle was very time consuming. In the future this would need to be provided by the vendor rather than added in after the fact. As an output of this project we hope to give a recommendation to designers so that this data product can be more effectively used in the future. Another project output might be a base outline for a standard component shapes library that could be used for development of these schematics.

One unexpected side effect of using SVG format is the length of time it takes to parse large SVG images. As we added additional content to the image in order to change component colors or store component specific information the load time increased. The length of time it took to resolve the image after a zoom in/zoom out also increased. Work will need to be done to mitigate this issue or perhaps other formats would provide better speed without a sacrifice of portability. Developing an SVG loader in Java may also enable faster load times.

8.2 Diagnostic Models

Diagnostic model development is performed by an entity (e.g. contractor, engineering, etc) that is not part of the ACAWS project. The development of the diagnostic models should include inputs from the ACAWS project so as to ensure maximum reuse, efficiency, and thus cost savings.

For the purpose of the demonstration scenario, the diagnostic model used for ACAWS was the Ares I Thrust Vector Control (TVC) system developed in TEAMS Designer. Because the system had already been modeled, we realized the time savings in not having to develop an entirely new model. However, we also found that several aspects of the model were not suitable to the scenario we had chosen, and additions had to be made to the model. Namely, the issues we encountered were:

- The Ares I model did not represent the control system or the electrical power system of the TVC, only the mechanical portion. In terms of actionable procedures that a flight controller might follow after a failure in the TVC subsystem, most of those are concerned with command and control failures rather than mechanical failures. For the purpose of our demonstration scenario, we added upstream components to the model that could fail and cause a break in the communication path of signals from the commanding GPC. Additional tests were also added to the model in order to implicate these components.
- Failure modes in the model were based on the Failure Mode Effects Analysis (FMEA) failures. This allows greater traceability back to Constellation products but is also limited to a smaller subset of potential failures from what could potentially be diagnosed using model based diagnosis techniques. The FMEA failure mode names were also overly long and often too generic to indicate a particular point of failure. One example that pertains to our scenario is when the Secondary Delta Pressure sensor goes out of limits and the model is unable to isolate to the specific actuator that has failed.
- The wrapper code tests that had been developed for the runtime diagnosis of the TVC were based on prelaunch phase. A new series of tests had to be written to cover a different set of thresholds for ascent and also to correspond to the added components mentioned above. In the future, any TEAMS models provided to the Elements as deliverables would also need to specify a method of evaluating tests in the model based on phase of flight and/or current configuration.
- An accreditation process for the TEAMS Designer model and D-matrix is beyond the scope of this report. One source is to reference the Ares 1 FFA Accreditation Process [FFA Cycle1B Report].[38]

8.3 Data System

The data system developed for the demo was comprised of C/C++ code and is much simpler than the system that would be put into place for a fully scaled implementation. Most of the lessons learned for future implementation relate to the integration with MCT.

---

[38] Please contact Peter.I.Robinson@nasa.gov for a copy of the FFA Cycle 1B report.

In the future, the data for ACAWS would be fetched from the SITF server through MCT, obviating the need to write an additional interface.

8.4 User Interface

There were a number of lessons learned during development of the user interface, pointed out to a large degree by the system experts who were familiar with existing flight control software and capabilities. The following is a list of decisions made that will most likely be carried forward from the current design:

- The health annunciator panel distinguishes between a component that has failed and one that only contains failed components beneath it. If the entire component is bad the label is turned solid red, but if lower-level components have failed the label is given a red outline.
- Need to make it evident to the operator that additional failures may have occurred within the annunciator panel, even after the tab has already turned red. This led us to change the behavior of the tab to flash between red and blue after a change occurs within the panel, but more could still be done to draw the user's attention.
- Events that repeat often should not be displayed in the event log window. In particular, the secondary delta pressure exceeds a user-defined "soft" limit multiple times during the scenario, and we made the decision not to include that in the log. We also removed the associated test from the TEAMS wrapper code due to the questionable diagnosis that something has actually failed at that point.
- Because the health annunciator panel can be resized to a very small size, the possibility of a label turning red while it is not displayed exists. In order to alert the user, we added indications next to the scrollbar that rescale when the window is resized and indicate the location of each failed/degraded annunciator.
- It was decided that those diagnoses that have no procedure or flight rule associated with them should still give an indication of the lack of documents. The text now brings up "No Procedure" or "No Rule" if that is the case.
- The TEAMS diagnosis is currently based on a "Minimal diagnosis", which may potentially conceal some multiple-fault situations that are less probable but still possible. We also encountered naming convention issues when the component name in TEAMS was too long for a readable display in ACAWS. The naming conventions used were originally based on a Ares I data product requirement.
- Diagnosis items in the diagnosis window are colored according to their status. A diagnosis containing at least one "BAD" element is colored red, while groups of "SUSPECT" components are colored yellow. When the user selects any diagnosis item and focus goes to that window, the default behavior is for the selected item to be colored blue, thereby obscuring the original color of the item. We experimented with several other color schemes but there was no resolution to this issue.
- Purple numbers on the display are shown to indicate that the values associated with a component are no longer valid. In our scenario, this includes all values going through the FA1 MDM after the first failure. There was an issue with purple on the light gray background that made it difficult to see the color difference, especially with discrete values represented with only a zero or one. We change the formatting of the font to bold to make this more evident.

- Currently there is no real notion of a diagnosis or event being “suppressed”, as is possible with a number of flight console tools today. More work will need to go into how this would be done, and what it means for a TEAMS diagnosis to be “suppressed”.

Appendix A: ACAWS Team

Management Team	Role	Affiliation
David Korsmeyer	IVHM Project Manager	NASA ARC
Ann Patterson-Hine	IVHM Principal Investigator	NASA ARC
Alan Crocker	ACAWS Task Customer	NASA JSC

Development Team	ACAWS Role	Affiliation
W. Jason Helms	GNC Subject Matter Expert	NASA JSC
Charles Lee	Schematics, data system	SGT @ NASA ARC
Sotirios Liolios	Domain expert	NASA JSC
John Ossenfort	Diagnostic models, user interface	SGT @ NASA ARC
Peter Robinson	Diagnostic models, schematics, data system	NASA ARC
Lilly Spirkovska	Task lead, interaction designer	NASA ARC

TEAMS Team	ACAWS Role	Affiliation
Somnath Deb	TEAMS expert	QSI
Sudipto Ghoshal	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
CAD	Computer Aided Design
CHIT	Abbreviation used for Mission Action Request
ConFRM	Constraints and Flight Rule Management
CRANS	Configurable Real-time Analysis System
CSV	Comma Separated Values (common file format)
CxPASS	Constellation Procedures Application Software Suite
DFT	Design for Testability
DTO	Development Test Objective
ELOG	Event Logger
EPS	Electrical Power System
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)
GNC	Guidance, Navigation, and Control (controller position for Shuttle)
IMS	Inductive Monitoring System (aka AMISS at JSC)
ISP	Information Sharing Protocol
JPL	NASA Jet Propulsion Laboratory
JSC	NASA Johnson Space Center
KSC	NASA Kennedy Space Center
LCC	Launch Commit Criteria
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)
MSFC	NASA Marshall Space Flight Center
MSID	Measurement Stimulation Identification (term used for to identify a specific measurement)
MSK	Manual Select Keyboard (Apollo); now refers to tabular display.
ORU	Orbital Replaceable Unit
PRACA	Problem Reporting and Corrective Action
PTT	Part-task Trainer
QSI	Qualtech Systems Inc.
RECON	Reconfiguration
SFRM	Space Flight Resource Management (similar to aircraft Crew Resource Management, CRM)
SITF	Source-Independent Telemetry File
SRB	Solid Rocket Booster
TEAMS	Testability Engineering and Maintenance System
TVC	Thrust Vector Control
Unique-identifier	PUI, MSID, CUI, etc.; a method to associate a parameter/measurement with a unique name
VV&A	Verification, Validation, and Accreditation

Appendix C: ACAWS Framework Design

The following figures show the initial design of the ACAWS user interface. The design was modified because of limitations of the implementation software. The block diagram was modified because of user input.

Figure 31: ACAWS application window layout.

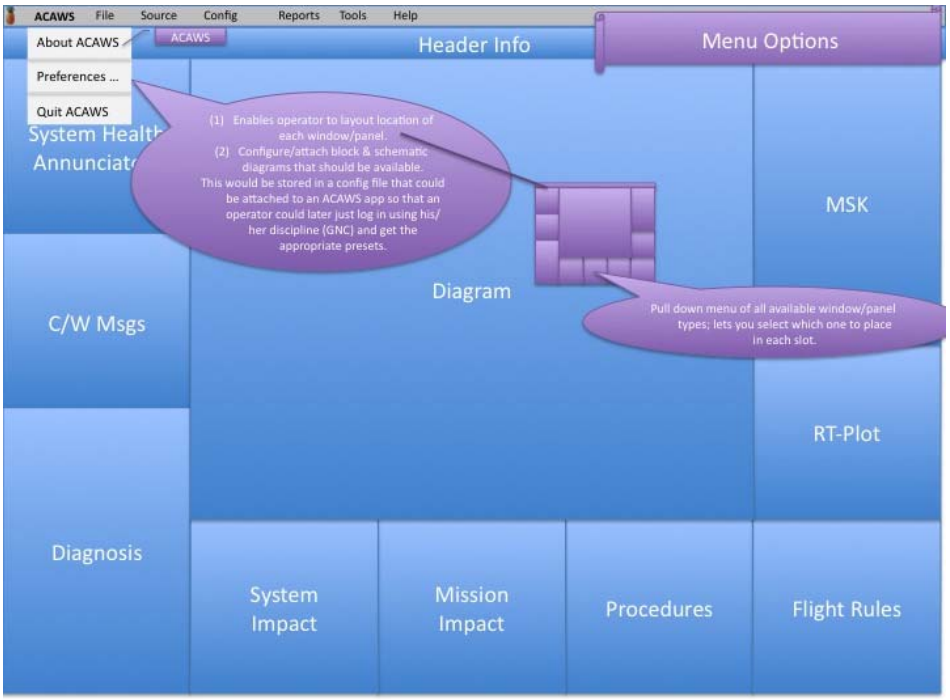


Figure 32: Window layout preferences details.

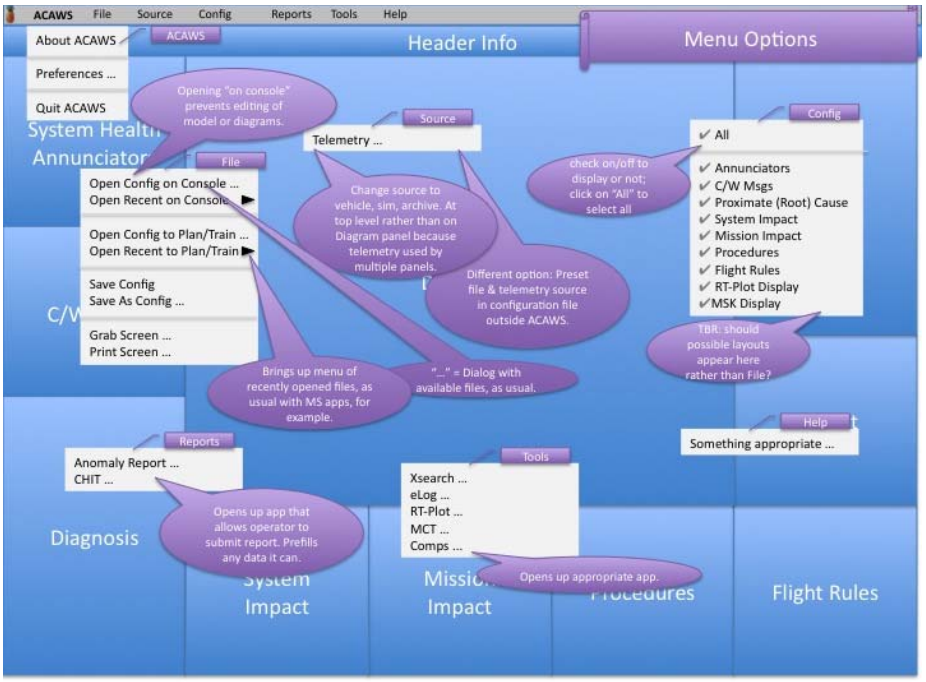


Figure 33: Window layout details.

Figure 34: Window header details.

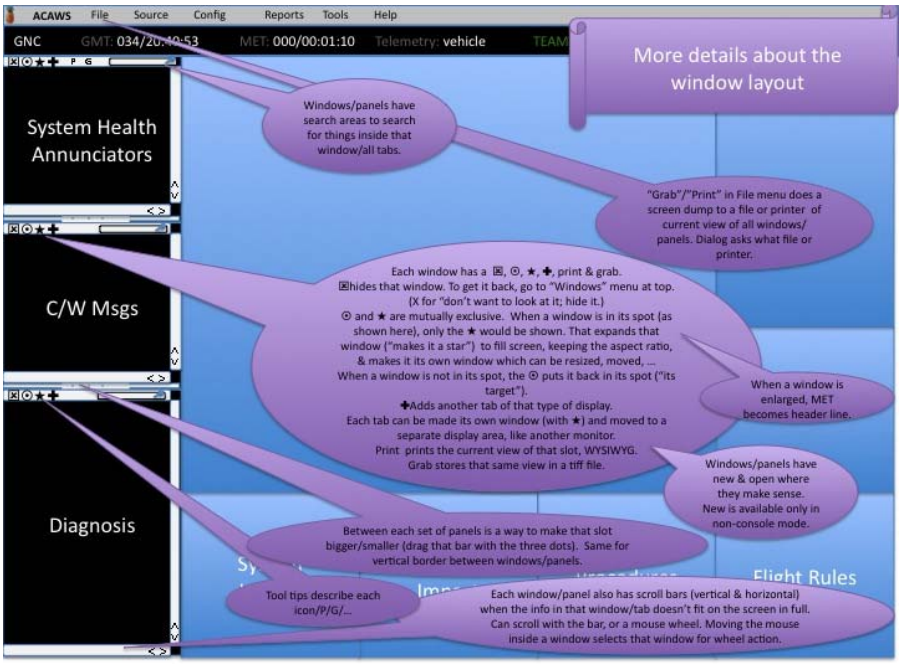


Figure 35: Panel details.

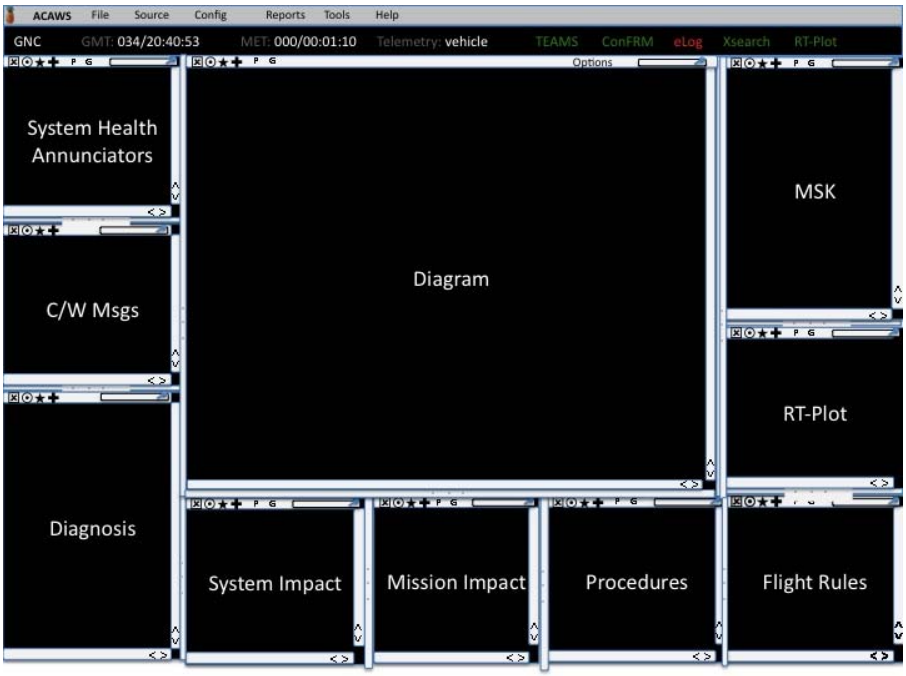


Figure 36: All panels shown.

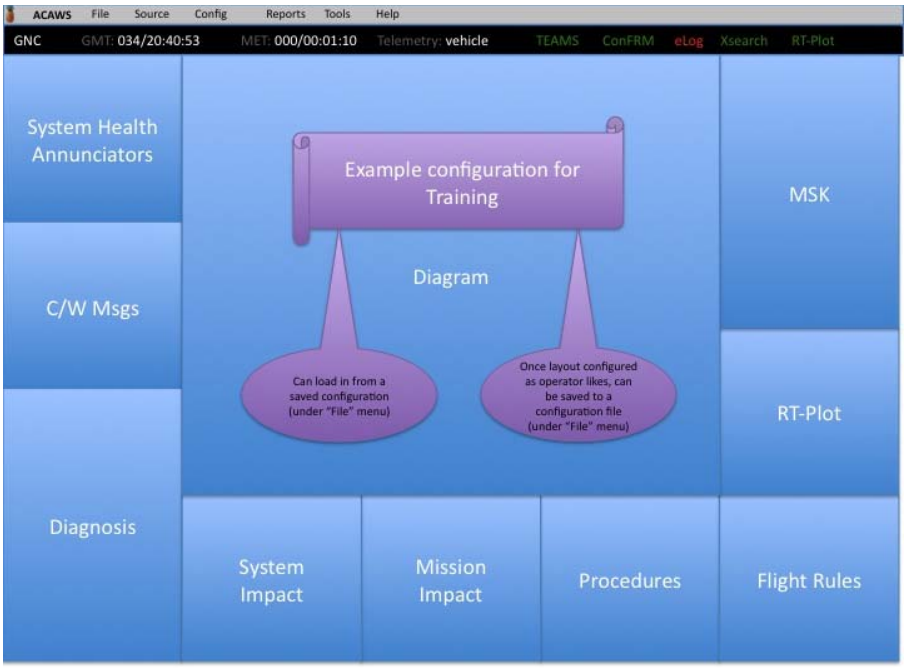


Figure 37: Window configuration for training (example).

Figure 38: Window configuration for troubleshooting (example).

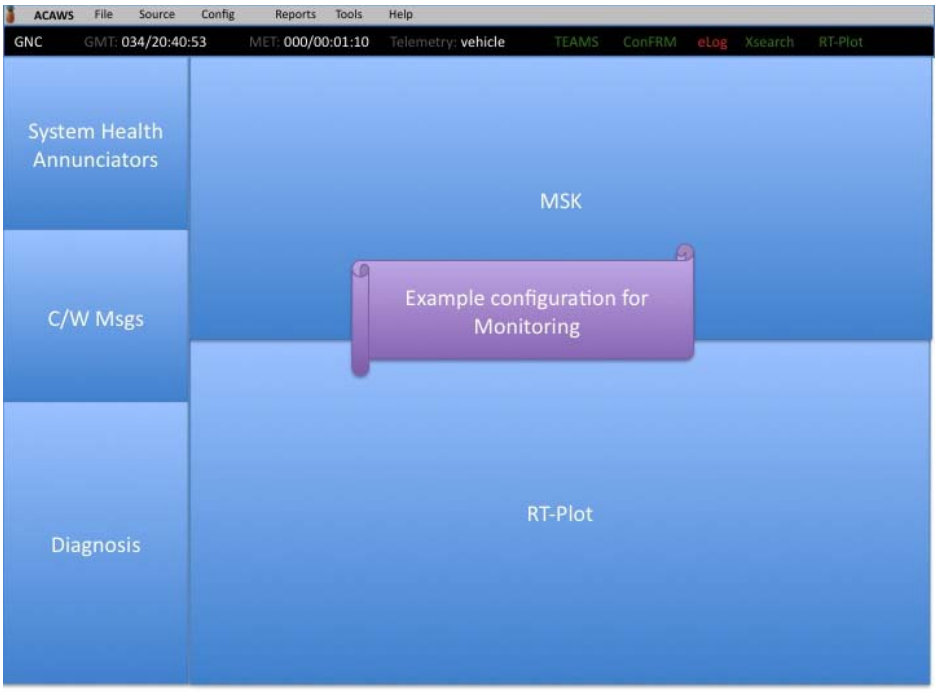


Figure 39: Window configuration for monitoring (example).

Figure 40: Window configuration for analysis (example).

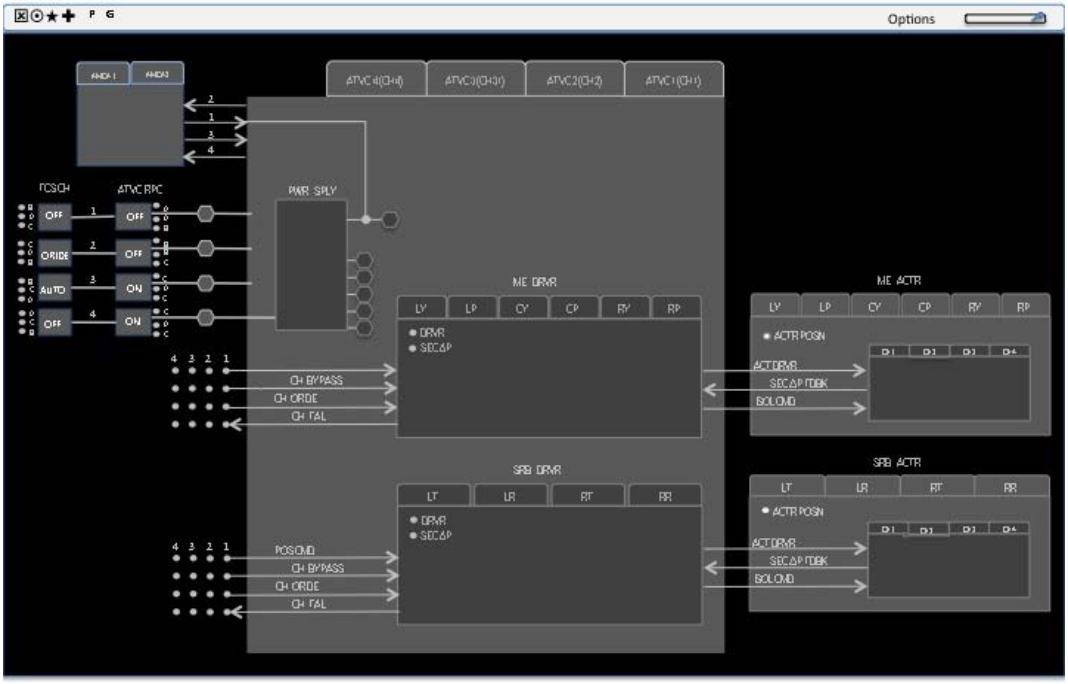


Figure 41: Block diagram design.

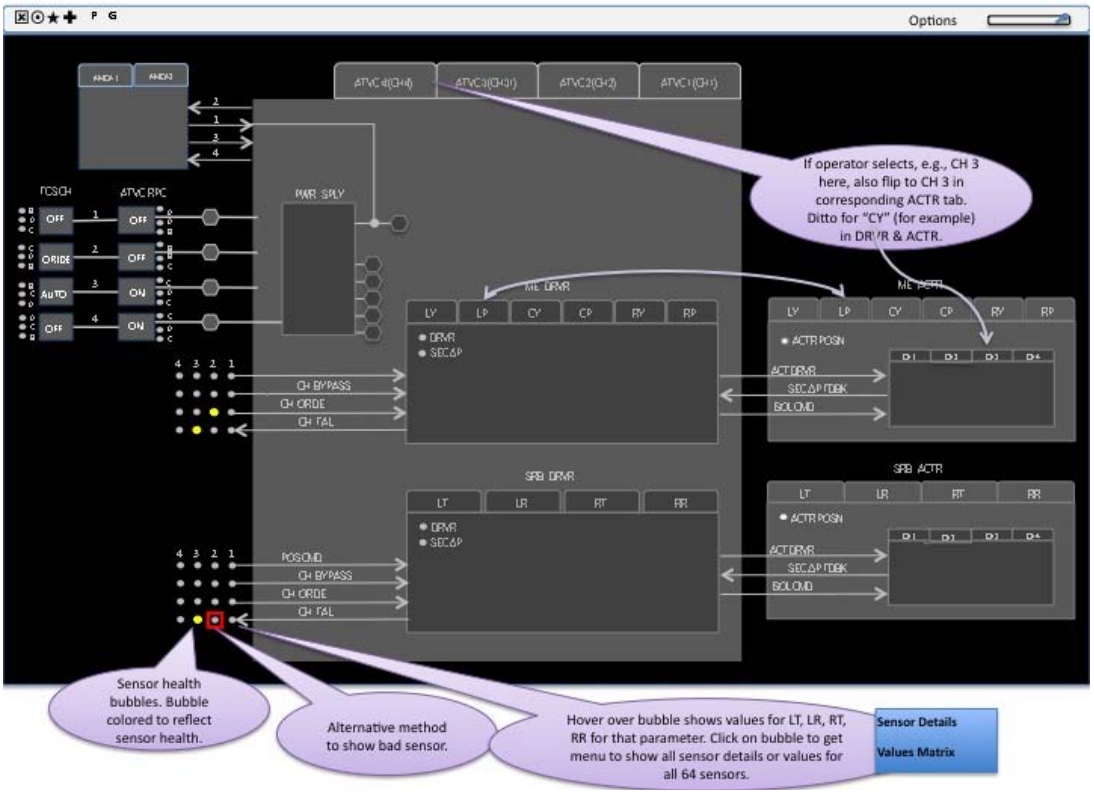


Figure 42: Block diagram details.

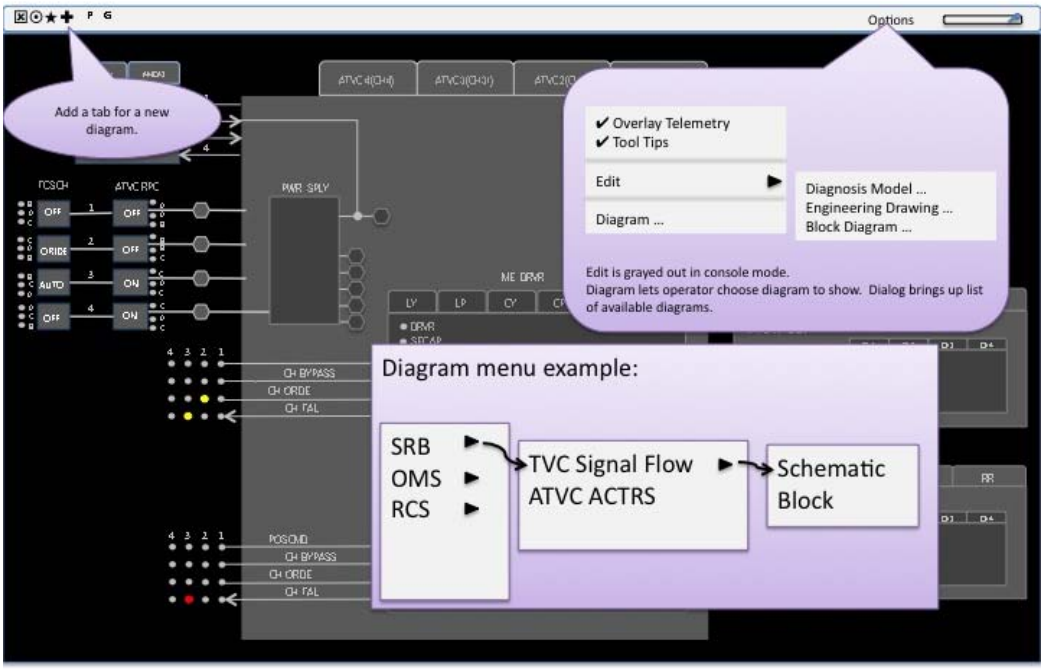


Figure 43: Block diagram menu details.

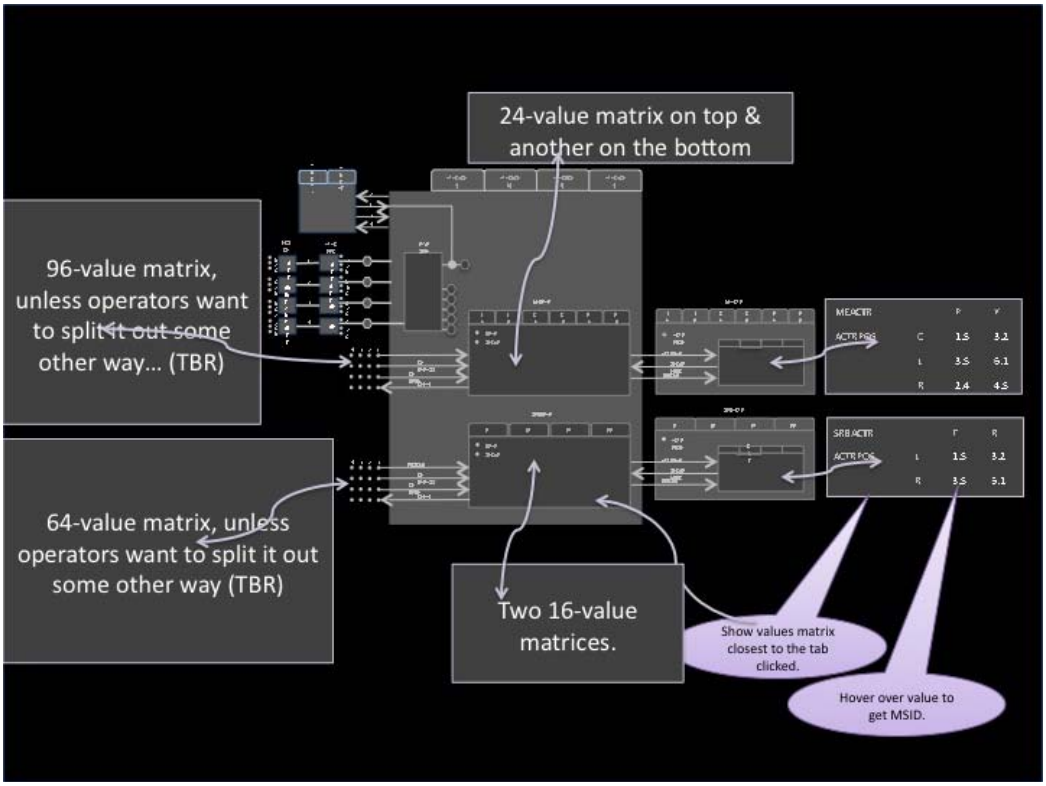


Figure 44: Block diagram values matrix details.

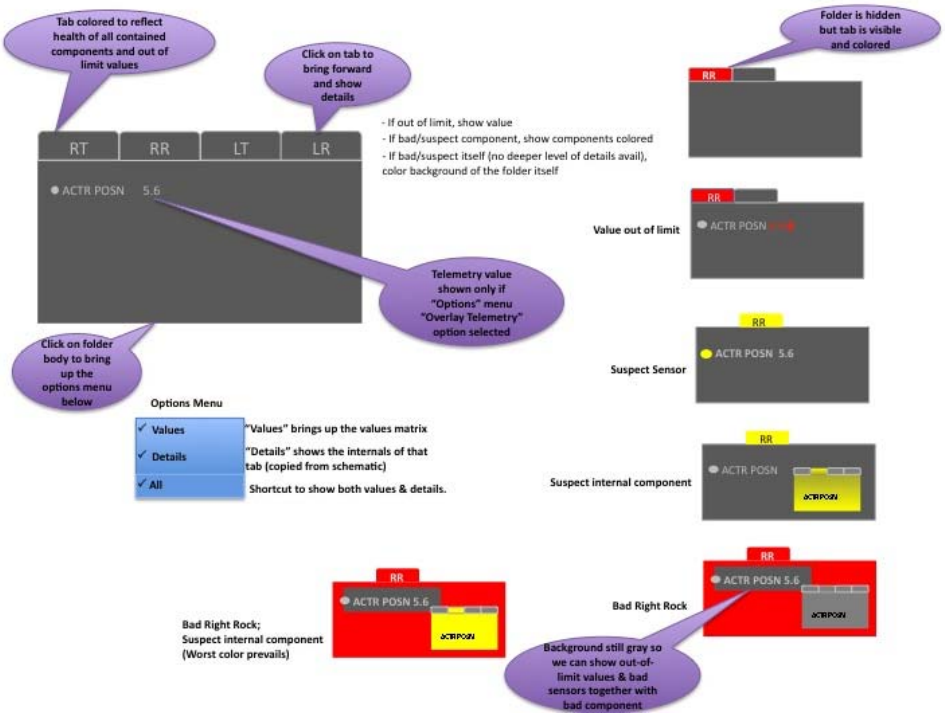


Figure 45: Block diagram folder details.

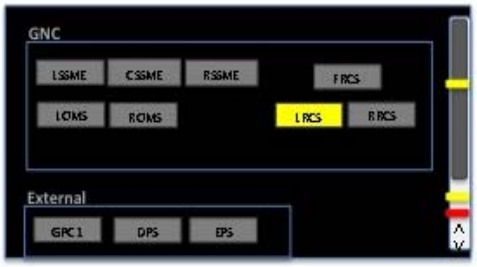


Figure 46: System health annunciators panel.

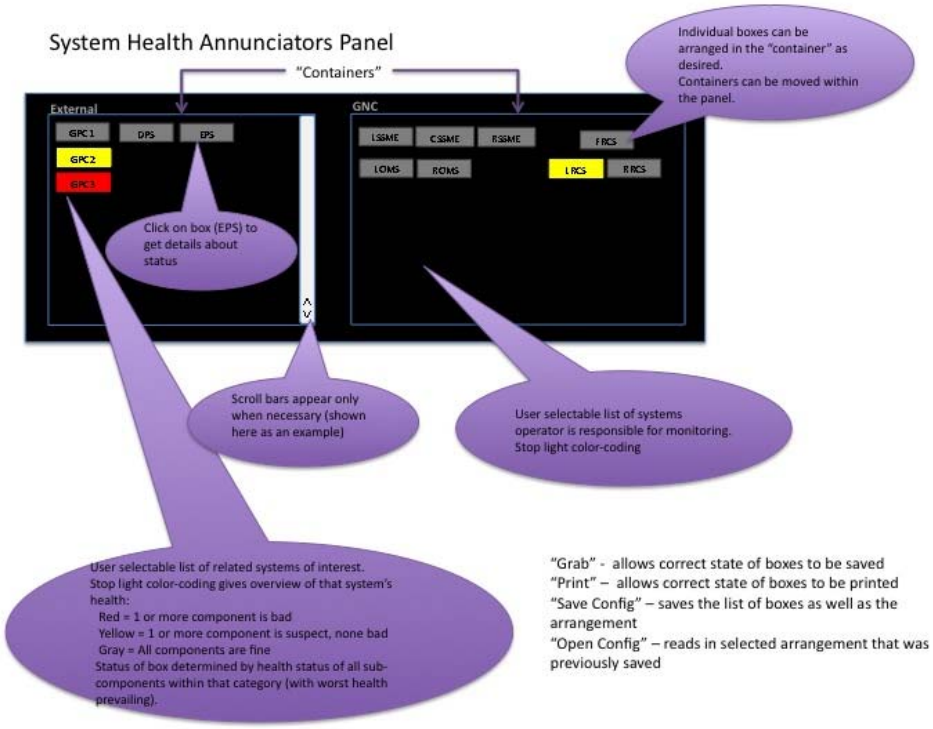


Figure 47: System health annunciators panel details.

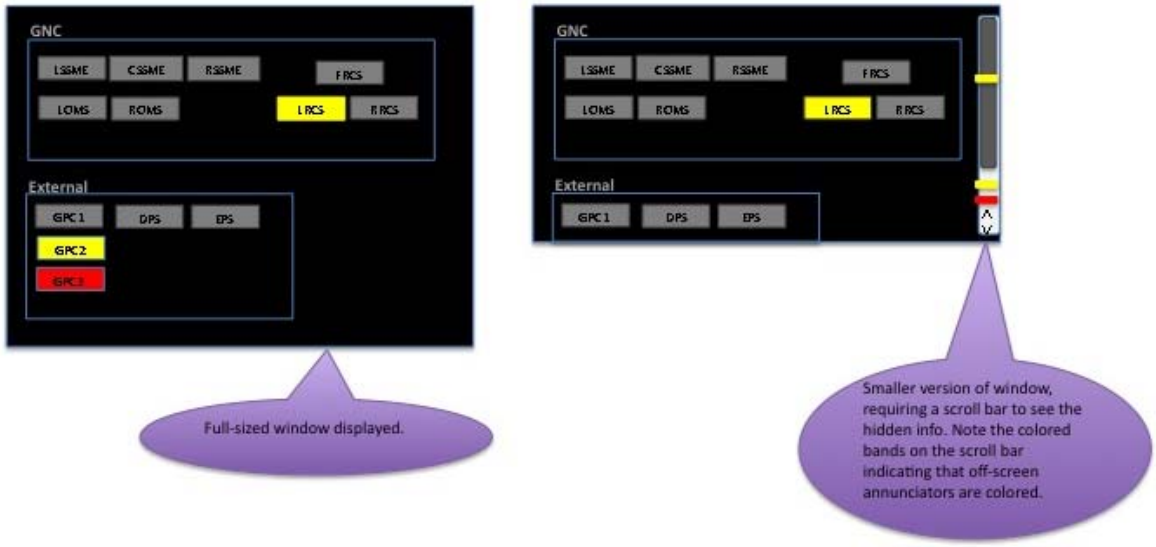


Figure 48: System health annunciators panel scroll window details.

Figure 49: ELOG messages panel.

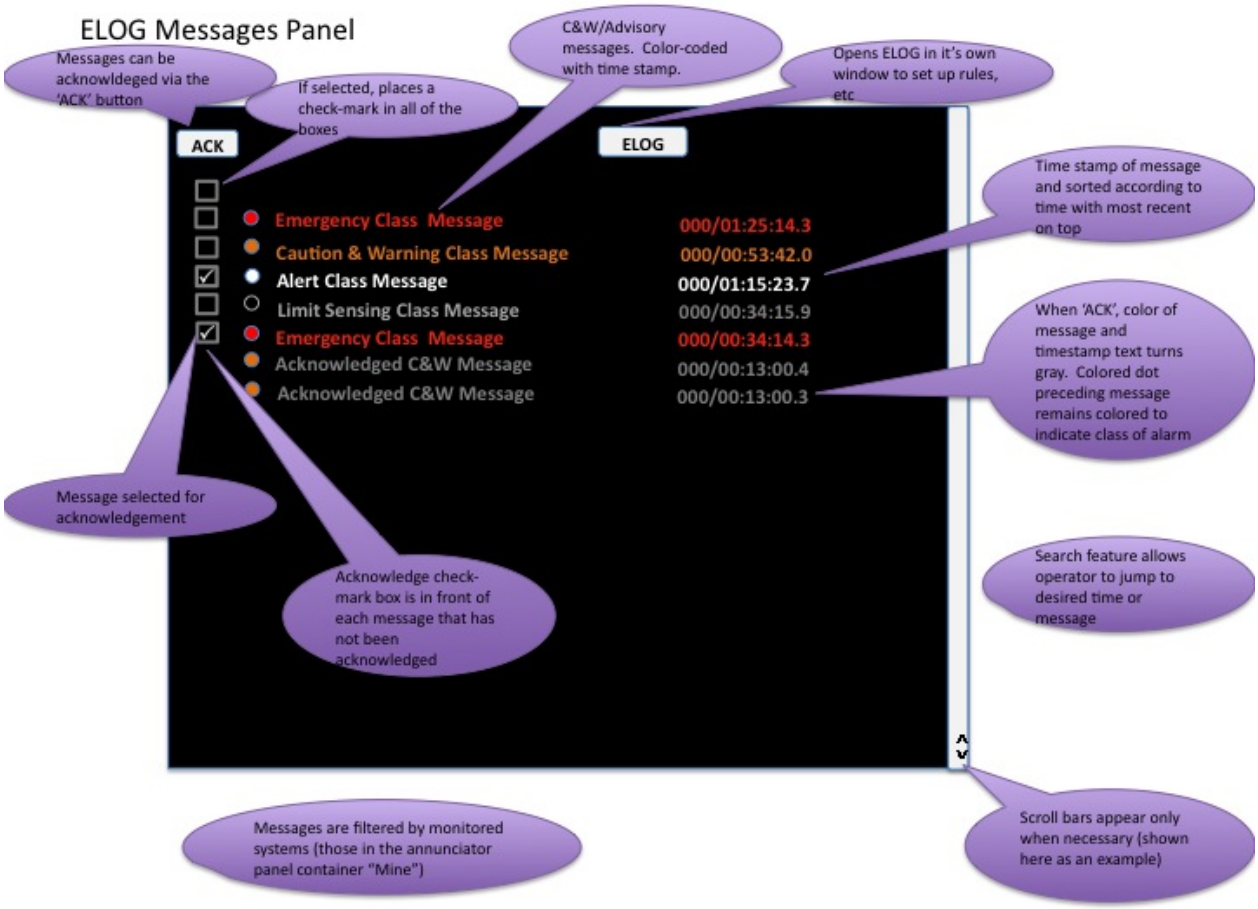


Figure 50: ELOG messages panel details.

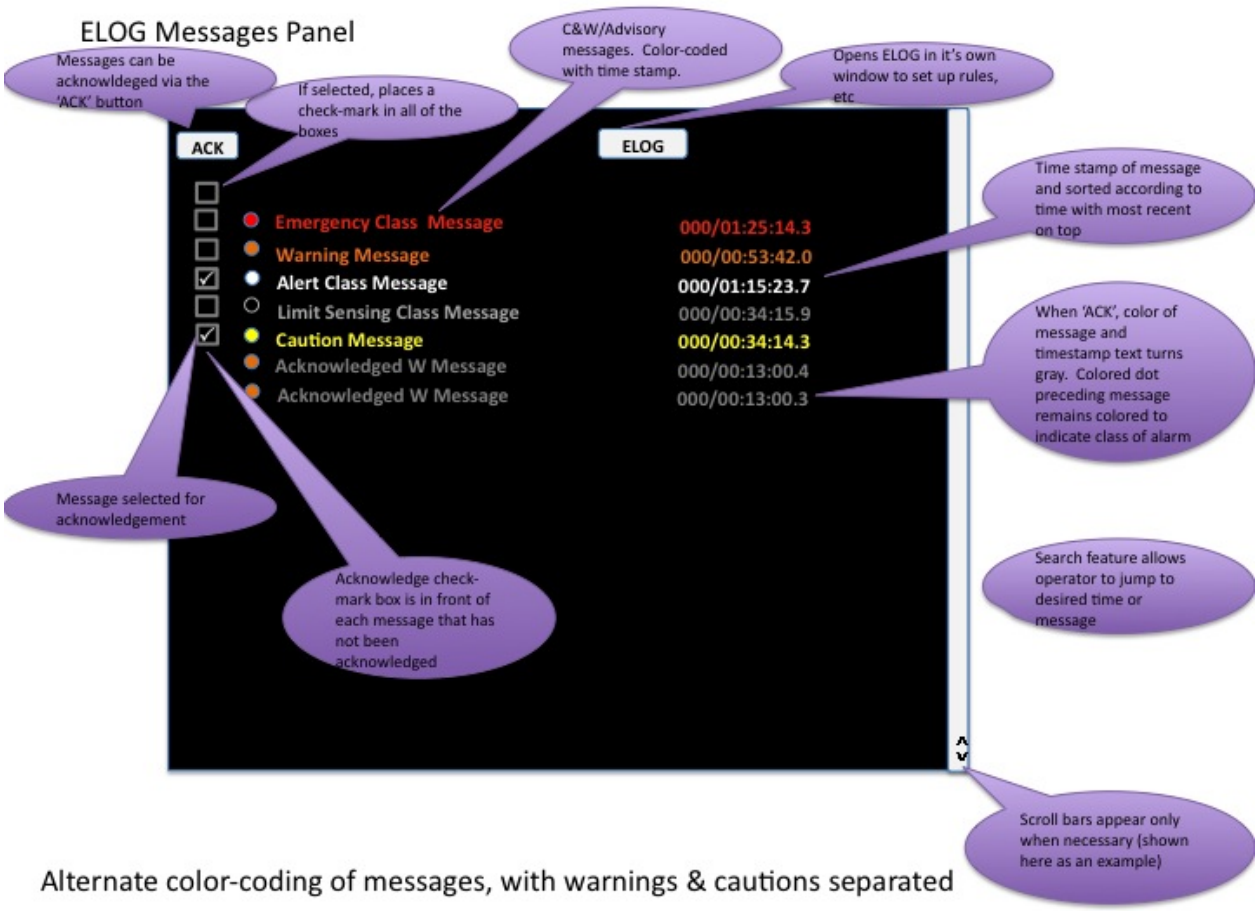


Figure 51: ELOG messages panel details, alternate color-coding.

Figure 52: Procedures panel.

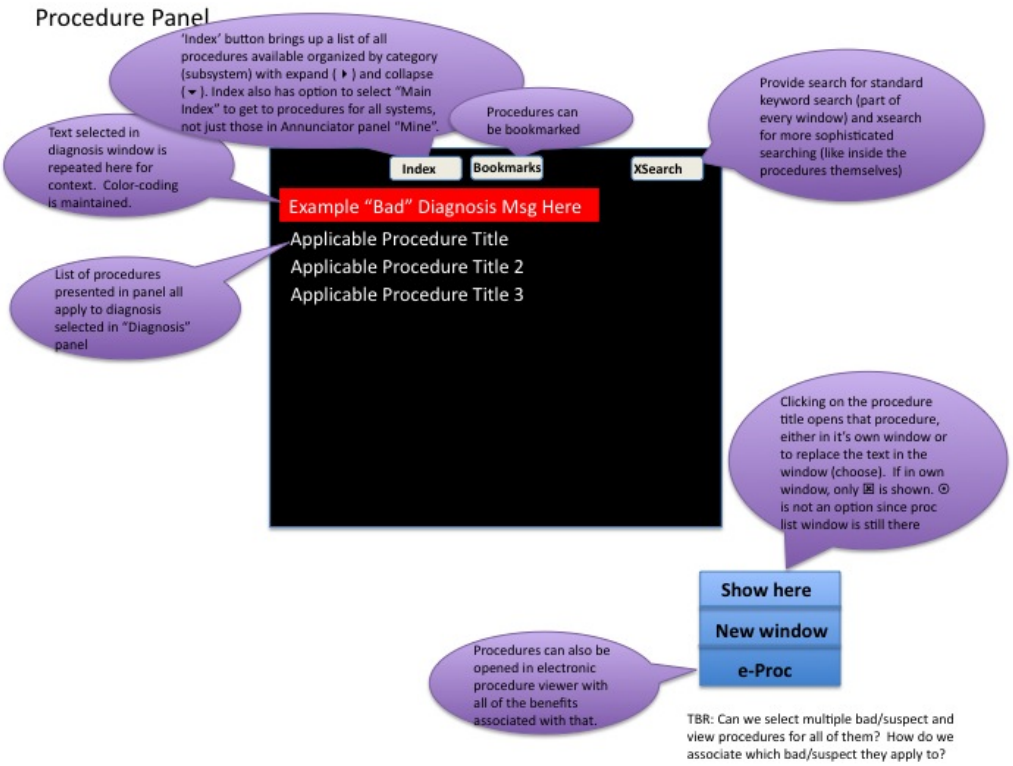


Figure 53: Procedures panel details.

Figure 54: Flight rules panel.

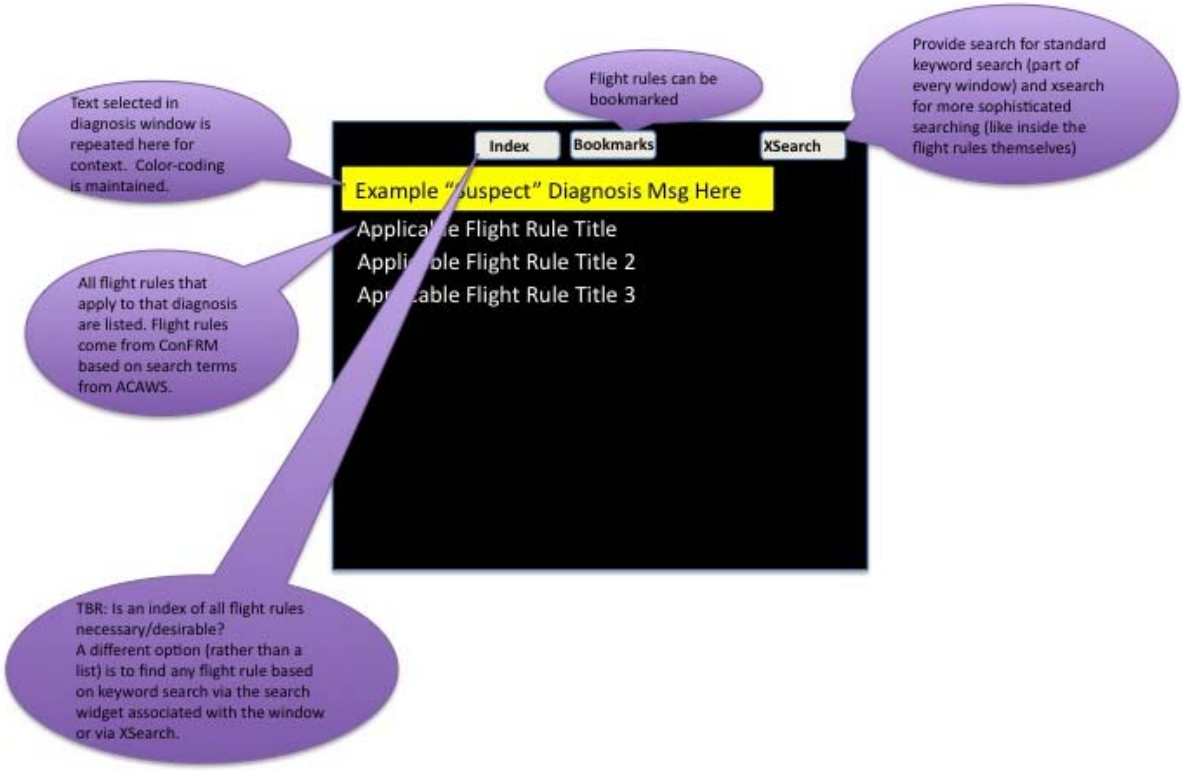


Figure 55: Flight rules panel details.

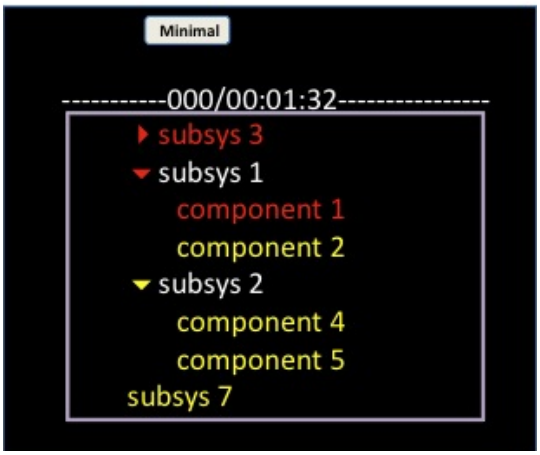
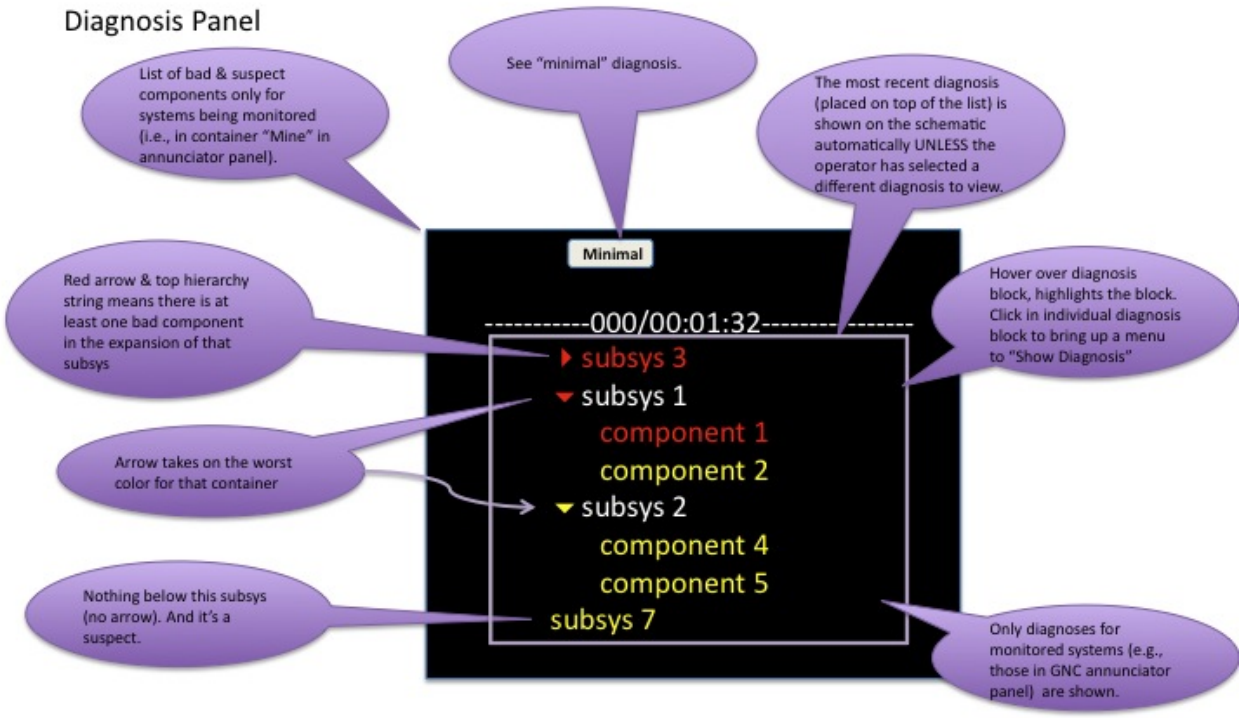


Figure 56: Diagnosis panel.



TEAMS is continuously updating the diagnosis based on current conditions. If conditions remain such that the same diagnosis is computed, only the initial diagnosis is shown. Duplicate diagnoses are not shown. When a different diagnosis is computed (based on a different set of conditions detected), it is shown. The original diagnosis (if it occurs again) would then be shown again if the original conditions reappear. That is, you can get diagnoses: A B C A, but will not get diagnoses: A A A A B C A A A

Figure 57: Diagnosis panel details.

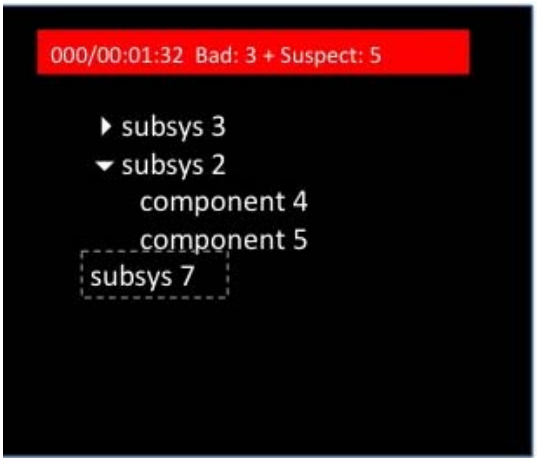


Figure 58: Systems impact panel.

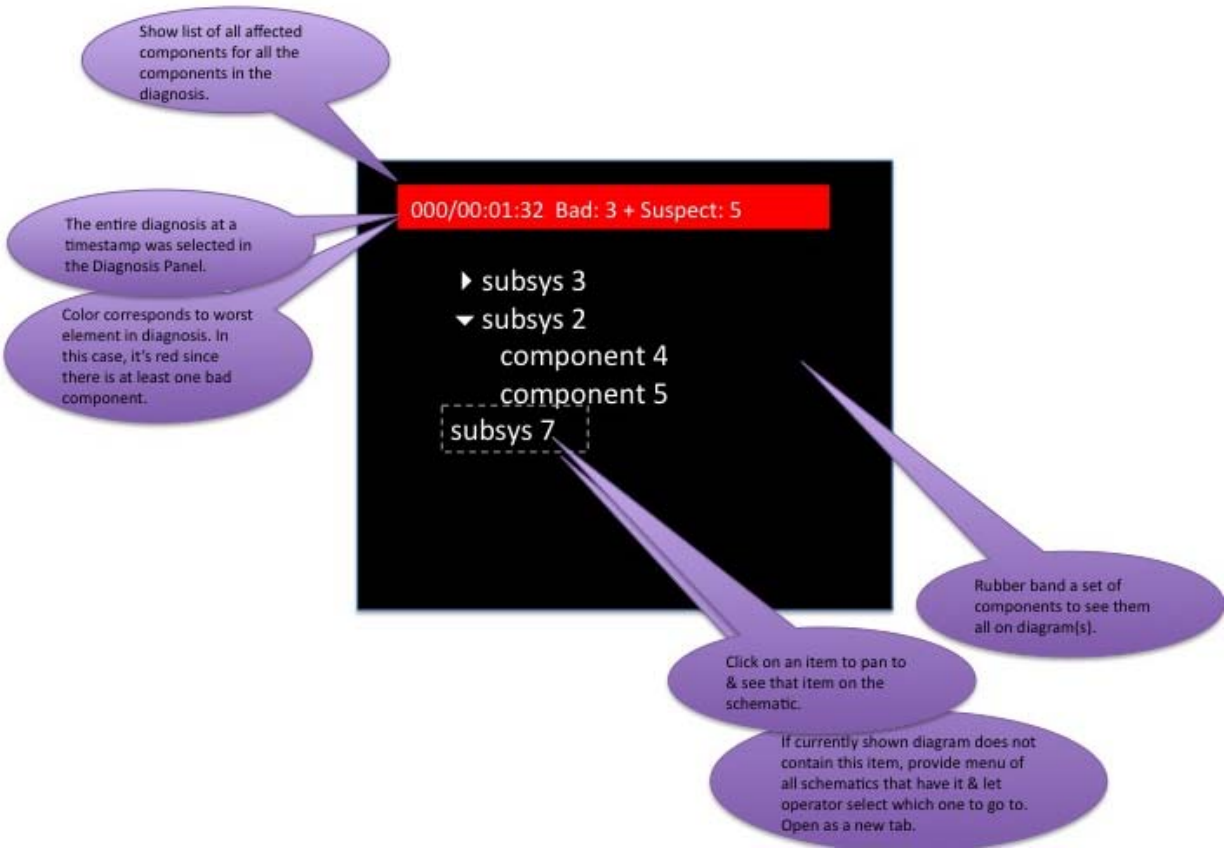


Figure 59: Systems impact panel details.

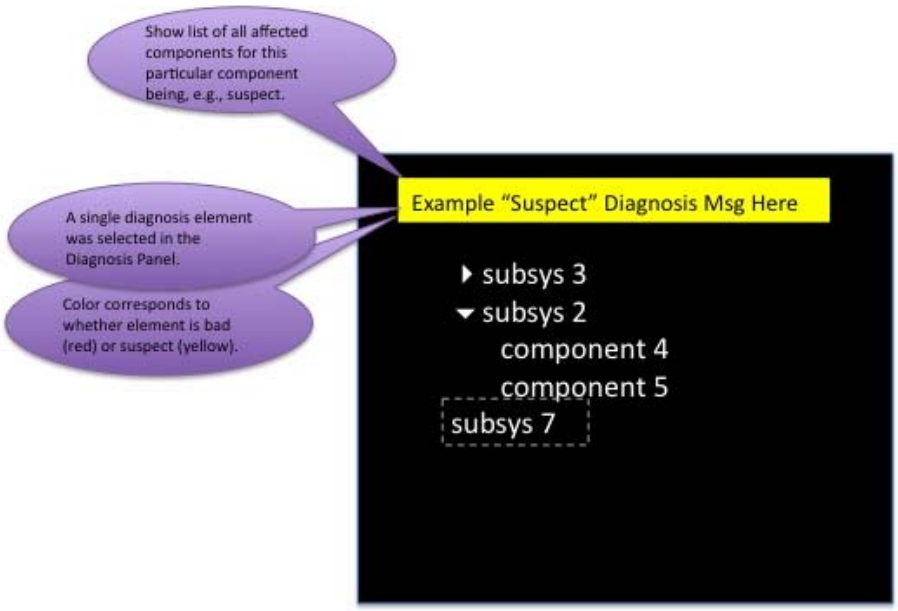


Figure 60: Systems impact panel, more details.

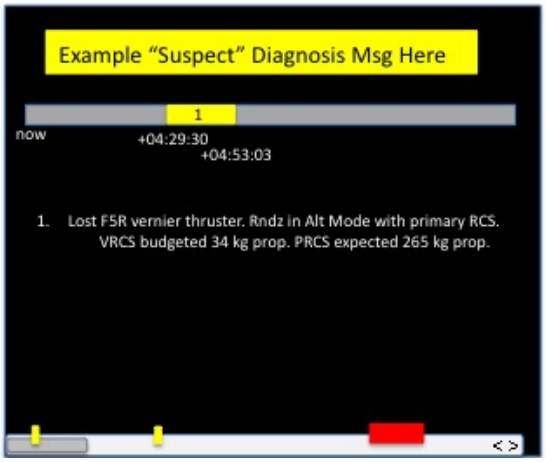


Figure 61: Mission impact panel.

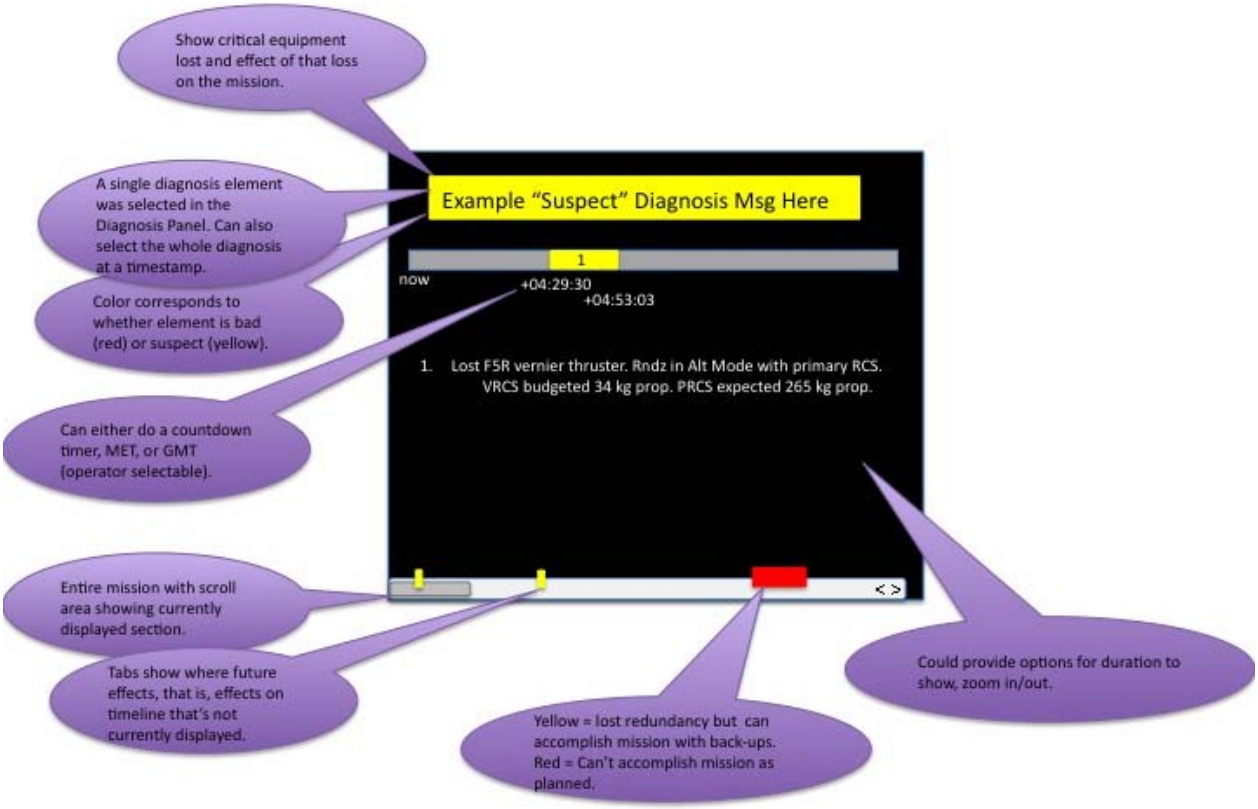


Figure 62: Mission impact panel details.

Figure 63: MSK tabular display panel.

Figure 64: Trend plot display panel.

Appendix D: Notes

The following notes are some ideas that the team considers important enough to make note of for future discussions. The notes contained reference back to the text; however, those references were bungled over the many versions of this report.

- Should consider modification of ACAWS data sets concurrently by multiple users. Integrating those modifications into a single data set will likely require human collaboration. It becomes especially difficult when multiple people are independently modifying the same component. Merging of model edits from multiple users is as complex as merging software code modifications. Things are slightly easier if I/O ports are not changed. Implies centralized repository of models, perhaps in RDS.
- Need to evaluate the benefits and drawbacks of connecting to ISP via MCT versus a direct connection to ISP null server. Some possible benefits of connecting via MCT include existing infrastructure for displaying/manipulating visual elements, ISP connectivity, and transparency of telemetry value source.
- Extracting a functional model from physical model is a difficult thing to do.
- Ability to view health status on the schematic is complicated because the sets of good, bad, and suspect can represent many actual state vectors of the system. Thus, the overlay of the info can be misleading. Producing consistent diagnoses from the three lists (good, bad, suspect) is not trivial.
- To clearly distinguish modification of a model, need to define a model diff function which can take two models are determine the changes.
- The models-to-D-matrix relationship is many-to-one. That is, many models can translate to a single TEAMS D-matrix. Because of this, model changes will need to be constrained to changes of the graphical representation of the model and cannot be made directly in the D-matrix or tabular representation. Changes to the D-matrix cannot be uniquely translated back to a graphical model.
- Enumerate what entities MCC operators may need to modify for each of the plan-train-fly tasks. Of this list, enumerate the entities that can be modified in ACAWS vs. those that must be done in TEAMS Designer.
- Should MCC operators be allowed to change the relationship between tests (observable effects) and failure modes?
- Formally map every piece of TEAMS model: (module, switch, AND node, test point, and effect node) to source material. That linking is critical to the review process as well – might as well use the information during operation.
- Showing spatial information (like physical location of heaters) can be implemented in multiple ways. One is to show a 2D view of the information, as they do now on system handbooks drawings. The other would require 3D drawings. The 3D approach would require mappings between 2D schematics and 3D drawings, a non-trivial process. It's also not trivial to compute distance between components to show spatial relations.
- Different types of zoom: physical, functional and Level of Detail (LOD).
- The structure of ACAWS will determine how difficult this is to implement. If the model has a physical breakdown then extracting functional breakdown will be hard.
- A long-term requirement may be to subscribe to DVIS (old) VOIP voice loops to get context information to focus models. This would anticipate the controller's need for a model based on what's being talked about on the voice loop. Many issues to resolve if this becomes a need, including which controller is focused on which systems, which types of conversations require looking at a new model or a new area of the model, whether users want displays to refocus without their input, etc.
- To search ACAWS models, entities will need to be labeled (tagged). In TEAMS, labels can be placed on modules, testpoints, tests, effect nodes and lines. Labels cannot be placed on AND Nodes and switches.
- Connecting paths between selected items may be different in nominal situations versus in failure situations. For example, if filter is clogged, flow backs up. The path between selected items may vary according to the fault case. TEAMS Model has to capture both the nominal and failure path. On Ares, when connectivity is reviewed some failure paths are given a tag of "model abstraction" because they are only in the TEAMS model to propagate faults upstream.
- A propagation path in TEAMS is obtained via DFT analysis. TEAMS Designer shows it as a purple line between components. A limitation of the TEAMS DFT analysis is that if there is not a path from fault to

test then no path is shown. It might be helpful to see a partial path to determine how far propagation occurred before it was stopped.

- Priority/severity information may be provided in Hazard Reports or other program documentation. ACAWS should reuse this information whenever available. Need to determine how to assign severity/priority when more than one failure is occurring at once.
- To view diagnoses as a time-series plot, the diagnoses can be treated as enums and display in step function form as digital circuits are displayed.
- When specifying new tests directly on the ACAWS model, ACAWS will need to update the model itself, update the subscription to the appropriate comp, and generate the comp.
- In the Ares FFA project, failures modes are grouped by using TEAMS hierarchy labels for LRU. All failure modes under an LRU are grouped with the LRU.
- To view suspect components in probabilistic order, need to define source material for model. Ranking by probability can be done if failure modes are physical. If failure modes are functional then ranking is not possible.
- To get root cause with TEAMS, closest functionality available is minimal diagnosis. However, that fails when multiple groups of suspects can explain all the test results (rather than just a single group). In that case, there is no "TEAMS minimum diagnosis". Instead, the groups are kept on the suspect list.
- To view all items that would fail for a specified unique-identifier (PUI, MSID, etc.), can look down columns in the D-matrix that use that unique-identifier in a test.
- In regards to "Ability to suppress previous failures so that new failures are more evident," TEAMS should continue with the knowledge that previous failures occurred (i.e., switches should not need to be reset in the model). It's the user interface that has to mark it as "previously failed" vs. "newly failed" to make the appearance of new failures more obvious.
- In order to compute failure propagation timing, reference materials will need to contain that information. If the initial TEAMS models are not already populated with that information, MOD will need to add it to meet this requirement. Once the times are in the model, TEAMS can annotate the propagation line ("the purple line") with times. To use it in ACAWS, we'd need an API to retrieve that information from TEAMS.
- As long as a hierarchy label can be attached in the model, perhaps additional properties may need to used in TEAMS to model criticality. Probably need to attach criticality outside of TEAMS.
- Determining impact of failure to a mission requires mapping from behavior/structure to functions. Will need Masterlogic Diagram functionality – similar to what was implemented in SeaClif. Can also be pulled from FMEA RBD criticality (e.g. Crit 1R2 – means need two to fail to lose function).
- When considering impact of failures, limit to "annunciate when one failure away" rather than "specify number of failures away for any failure" because of the expected redundancy limitations of future spacecraft.
- Recovery procedures can be linked to single failures. For example, for ISS, Caution & Warning tables provide a mapping between conditions and corrective action procedures. If there are multiple failures, recovery procedures can be prioritized by failure priority. To recover from all failures simultaneously would be outside the scope of preplanned contingencies. A reactive planning software system would be needed to develop a procedure to recover from multiple failures. To reuse existing procedures for this task, procedures may need to be annotated with goal statements, i.e., rationale for each step in a procedure and any constraints for sequencing the steps.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*
10-10-2020

2. REPORT TYPE
Technical Memorandum

3. DATES COVERED *(From - To)*
10/2009 – 9/2010

4. TITLE AND SUBTITLE
Advanced Caution and Warning System, Final Report — 2010

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Lilly Spirkovska, Peter Robinson, Sotirios Liolios, Charles Lee ,
John Ossenfort

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
425180.04.02.02

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-2013-216509

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 16
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The current focus of ACAWS is on the needs of the flight controllers. The onboard crew in low-Earth orbit has some of those same needs. Moreover, for future deep-space missions, the crew will need to accomplish many tasks autonomously due to communication time delays. Although we are focusing on flight controller needs, ACAWS technologies can be reused for on-board application, perhaps with a different level of detail and different display formats or interaction methods. We expect that providing similar tools to the flight controllers and the crew could enable more effective and efficient collaboration as well as heightened situational awareness.

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	94	**19b. TELEPHONE NUMBER** ***(Include area code)*** (443) 757-5802