From Diagnosis to Action: An Automated Failure Advisor for Human Deep Space Missions

Silvano Colombano*, Lilly Spirkovska*. Vijayakumar Baskaran**, Paul Morris*,
William Mcdermott*, John Ossenfort**, Anupa Bajwa*
*NASA Ames Research Center
**Stinger Ghaffarian Technologies Inc.

The major goal of current space system development at NASA is to enable human travel to deep space locations such as Mars and asteroids. At that distance, round trip communication with ground operators may take close to an hour, thus it becomes unfeasible to seek ground operator advice for problems that require immediate attention, either for crew safety or for activities that need to be performed at specific times for the attainment of scientific results. To achieve this goal, major reliance will need to be placed on automation systems capable of aiding the crew in detecting and diagnosing failures, assessing consequences of these failures, and providing guidance in repair activities that may be required.

We report here on the most current step in the continuing development of such a system, and that is the addition of a Failure Response Advisor. In simple terms, we have a system in place – the Advanced Caution and Warning System (ACAWS) – to tell us "what happened" (failure diagnosis) and "what happened because that happened" (failure effects). The Failure Response Advisor will tell us "what to do about it", "how long until something must be done" and "why it's important that something be done" and will begin to approach the complex reasoning that is generally required for an optimal approach to automated system health management.

This advice is based on the criticality and various timing elements, such as durations of activities and of component repairs, failure effects delay, and other factors. The failure advice is provided to operators (crew and mission controllers) together with the diagnostic and effects information. The operators also have the option to drill down for more information about the failure and the reasons for any suggested priorities.

I. Introduction

The major goal of current space system development at NASA is to enable human travel to deep space locations such as Mars and asteroids. At that distance, round trip communication with ground operators may take close to an hour; thus, it becomes unfeasible to seek ground operator advice for problems that require immediate attention, either for crew safety or for activities that need to be performed at specific times for the attainment of scientific results.

To achieve deep space human missions, major reliance will need to be placed on automation systems capable of aiding the crew in detecting and diagnosing failures, assessing consequences of these failures, and providing guidance in repair activities that may be required. A system that provides some of these capabilities has been under development at NASA Ames Research Center for some years: the Advanced Caution and Warning System (ACAWS), which at this time combines dynamic and interactive graphical representations of spacecraft system, and uses a system model for diagnostic analysis and failure consequences analysis. (1,2) This system was tested on the Deep Space Habitat laboratory and recently with telemetry data during the recent EFT-1 flight.

We report here on the next development step for ACAWS. This is a system, called Failure Response Advisor (FRAd), that will provide advice regarding the *Severity* of detected faults and crucial timing factors for recommended repairs. The severity of detected faults depends on the *Criticality* of activities that

have been blocked by the faults. Severity and Criticality indicate specific distinct concepts that are explained and defined in the body of the paper. Development of FRAd adds a crucial automation step that would be necessary in a situation where ground support would be hampered by the long latencies expected on missions to Mars's vicinity and beyond.

II. System Architecture

The system architecture implements the following conceptual steps:
1. Faulty components and impacted components (the distinction is explained below) are identified by the existing ACAWS system. These components typically constitute resources that would be needed to perform mission activities (both human and automated).
2. The Failure Response Advisor determines what activities are blocked by the absence of these resources, together with their number and "criticality".
3. The number and criticality of these activities forms the basis for a computation of a "severity" value for each of the responsible faults.
4. Information on faulty components and suggested repair timings is provided.

Some concepts need to be clarified. The first is "fault" vs "impact". A component will be determined to have "failed" or have a "fault" ("faults" and "failures" are used interchangeably) by the diagnostic system based on the system model and incoming telemetry values. This implies that some function the component was designed to perform is no longer available. A fault can only be remedied by direct repair of the component or by exchange with a new working component. A faulty component will generally cause other components to cease functioning, for instance a faulty power supply may cause a light to be off. In this case we say that the light bulb has been "impacted" but has not "failed". Repairing the power supply will return the bulb to its full functionality (unless the light bulb coincidentally also failed). No repair or substitution is needed for the bulb. A typical diagnosis of a fault will in general be accompanied by a list of "impacted" components. These impacted components will not be in need of repair. Repair or substitution of the original faulty components will re-establish the functionality of all impacted components.

Other crucial concepts for our framework are those of "criticality" and "severity". These are often used interchangeably in the literature, but we found it important to use the two words in distinct ways. Criticality is associated with mission activities and, in our scheme (also dictated by common usage at NASA) it has a value that starts with "1" for activities that are crucial for astronaut survival and/or mission accomplishment, and may range to 3 or higher integers for less crucial activities, depending on how finely one wishes to classify other activities, such as selected science, spacecraft maintenance, comfort activities, etc. The criticality value of any given activity is specified *a priori* based on the mission designers' knowledge of the role of that activity in the mission. For example, the opening of a parachute at re-entry will usually be crucial enough for crew safety to merit a criticality of "1". A particular science experiment might be highly desirable, but not crucial to the overall mission, and thus may merit a criticality of "2" or "3", and so on.

Criticality is not associated with individual components. A component – or the components impacted by that component's failure – might be used by a number of activities of different criticality to the mission. Moreover, a component may be a necessary resource for a critical activity early in the mission, but its failure may not affect any activities later in the mission; thus, its loss of functionality would then be irrelevant. To accommodate the changing importance of components in different contexts during a mission, we assign a "severity" value to each failure. In this framework it becomes intuitive that a component failure, together with its impacted components, that blocks more activities of a given criticality than another component failure blocks will have a greater negative effect on the mission. Similarly a failure or impact that blocks an activity of high criticality has a greater negative effect on a mission than a failure that blocks an activity with lower criticality. We thus compare different faults according to the number and the criticality of blocked activities. We illustrate below how this severity value is computed.

The system architecture implements the above concepts according to the following block diagram (Fig. 1):

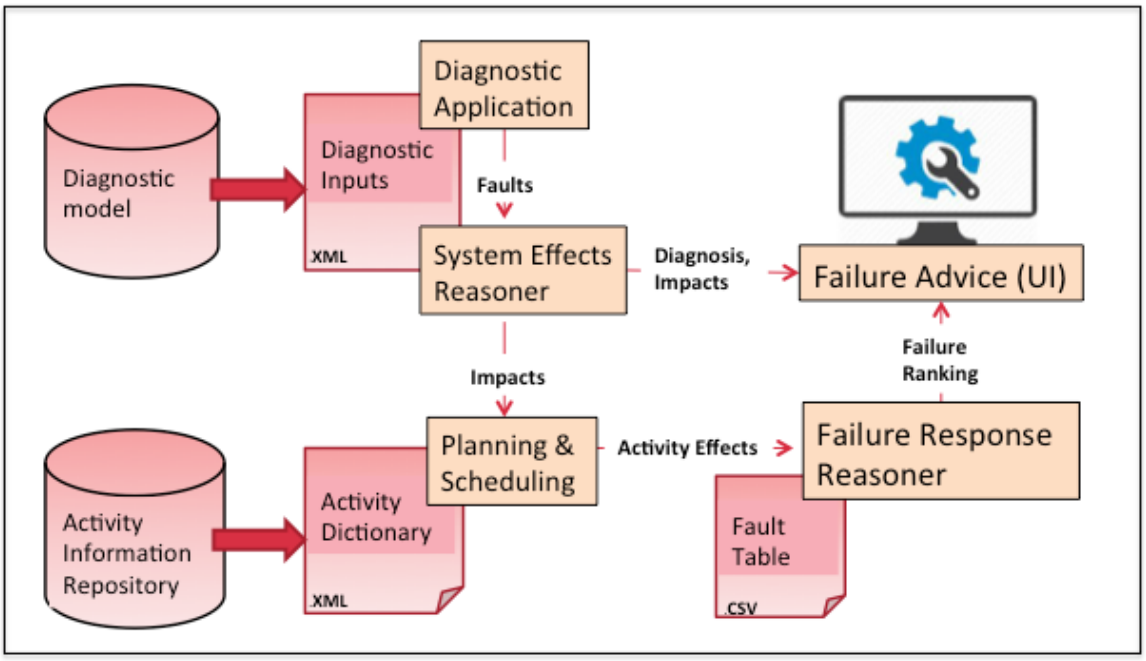

Figure 1. Flow Chart of Failure Response Advisor component data flows

From top left, ACAWS modules for diagnosis and system effects identify faults and impacts and deliver that information to the Planner. The Planner gathers information about activities from an Activity Dictionary. This information includes what resources are needed for given activities to be performed. The Planner correlates fault and impact information with its effect on activities to be performed and delivers the correlated information (Activity Effects) to the Reasoner (bottom right). The Activity Information Repository (bottom left), contains the activity information required by the Activity Dictionary and additional timing information required by the Reasoner. Additional information on faults is contained in a Fault Table (bottom right) and is delivered, as needed, to the Reasoner that computes fault severity, repair timing suggestions, and provides the advice to be published on the User Interface. The actions of all these components is explained in greater detail below.

III. Planning System

The planning system is based on a tool called SPIFe (3) that has been used extensively at NASA for activity planning (4). This has included scheduling for the International Space Station and Mars rover planning. SPIFe is used for plan creation, validation and editing. It reports on plan violations and relies on the user or other automated planner to repair these violations. These violations may include user-specified temporal constraints, over-subscribed resources, or failure to meet state requirements. The role of the planner in FRAd is to detect violations in the resources required to perform the set of planned activities. A failed or impacted component will typically be a resource for some planned activity, and the planner will check every activity, as stored in its Activity Dictionary, and will flag "activity violations" as a result of failed or impacted components that were needed as resources. The Activity Dictionary forms a static model of the application domain and allows specification of state variables, resource types, resource limits, and state requirements. For activities where violations have been identified ("blocked" activities), SPIfe will report timing information and Criticality Values (see section IV). Activities can also be pre-requisites for other activities, and pre-requisite activities are treated as a type of resource as well, which, if missing will block other activities. Note that a single fault can initiate a long cascade of consequences. First the fault will cause a chain of "impacts" among other components, as explained in II, then every faulty or impacted component will generally become a missing resource that will block the performance of planned activities.

If any of these activities are pre-requisite for other activities, those activities will be blocked as well. The planner in FRAd, determines the full list of blocked activities and sends it to the Reasoner together with the identity of the original fault that started the chain. Note that we are assuming that repair of the original fault (or faults) will re-establish the functionality of all impacted components and will allow the performance of all planned activities, by making all needed resources available again.

IV. Activity and Fault Information

One of the basic assumptions of this work is that there is information about activities and faults that can only be known *a priori* from an understanding of mission requirements and mission hardware and software. The role of FRAd is to identify the relevant information and to use it to compute the severity of faults and repair times. All activity information is stored in the Activity Information Repository. This includes timing information (Fig. 2), Criticality Value and an explanation for this value, to be displayed on the user interface if requested. It should be clear that all these values can only be known or estimated from an understanding of the mission that is the province of mission designers and systems engineers. For a real mission, mission designers and systems engineers would be required to provide this information. This is also how the Activity Dictionary used by the planner is constructed. In our case we use an even more comprehensive Activity Information Repository that informs both the Activity Dictionary and the Reasoner.

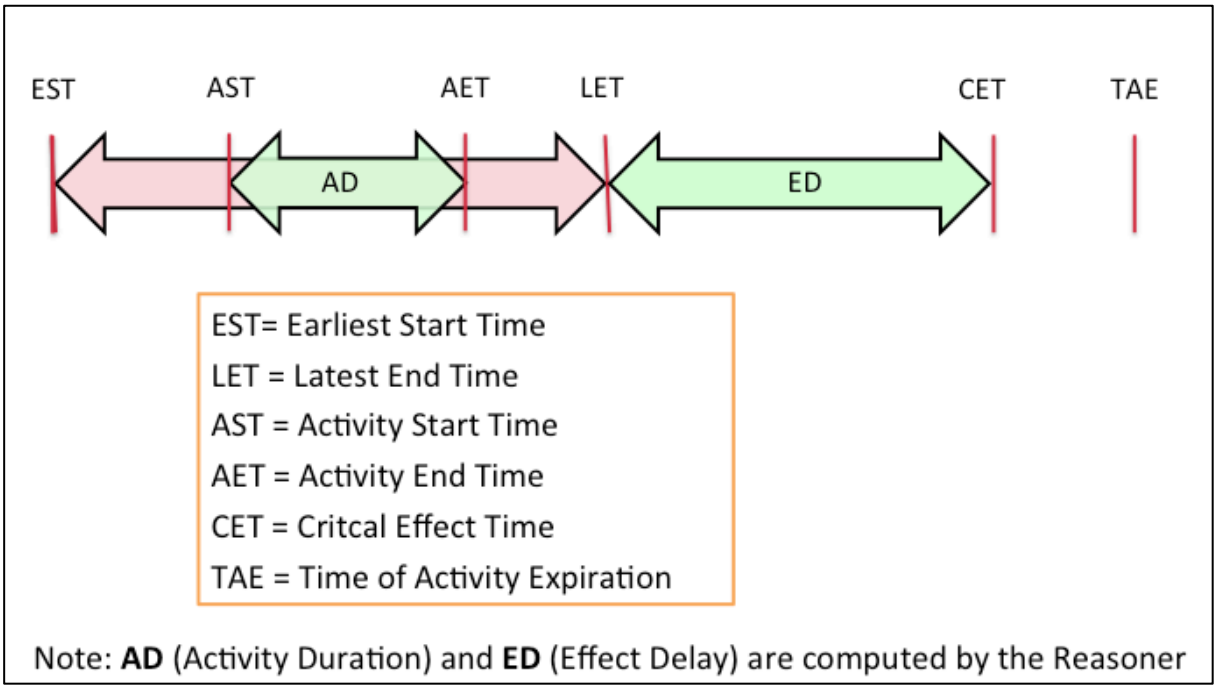

Figure 2. Activity timing elements required for the decision making process

Note that all values named "time" are points in time. Time intervals are called "duration" or "delay", and, in our system they are computed from the time values. Activities will typically be scheduled for a particular Start Time (AST) and End Time (AET), but there may be flexibility in the mission plan that will allow the activity to be started earlier, up to an Earliest possible Start Time (EST), and ended later, up to a Latest End Time (LET). The Critical Effect Time (CET), is the time when the consequences of a blocked activity would actually be registered by the system as an undesirable effect. For instance , the closing of valve may have been prevented by a faulty component. The consequence of this blocked activity would be the loss of a fluid, but the system might not be affected until the fluid is actually depleted some time later. The Time

of Activity Expiration (TAE) is a time when the activity has become obsolete and, even if repairs are accomplished, the activity is no longer needed. For example, imaging of an asteroid planned during a fly-by would be blocked by a power supply failure. Once the spacecraft flies past the asteroid and beyond the ability of the imaging system, even if the power supply is repaired, the activity is obsolete.

All these timing elements are required to decide if and when repairs will be possible and/or needed, and will serve as constraints for the planning system to schedule repair activities and re-schedule previously planned activities as needed. For this implementation of the system the advice provided is only based on the overall criticality of blocked activities and initial timing constraints, without further invocation of the planner. Closing of the cycle with re-planning is left for future system development.

Similarly to activity information, fault information consists of elements that will be known *a priori* by mission designers and systems engineers. These are shown in Fig. 3.

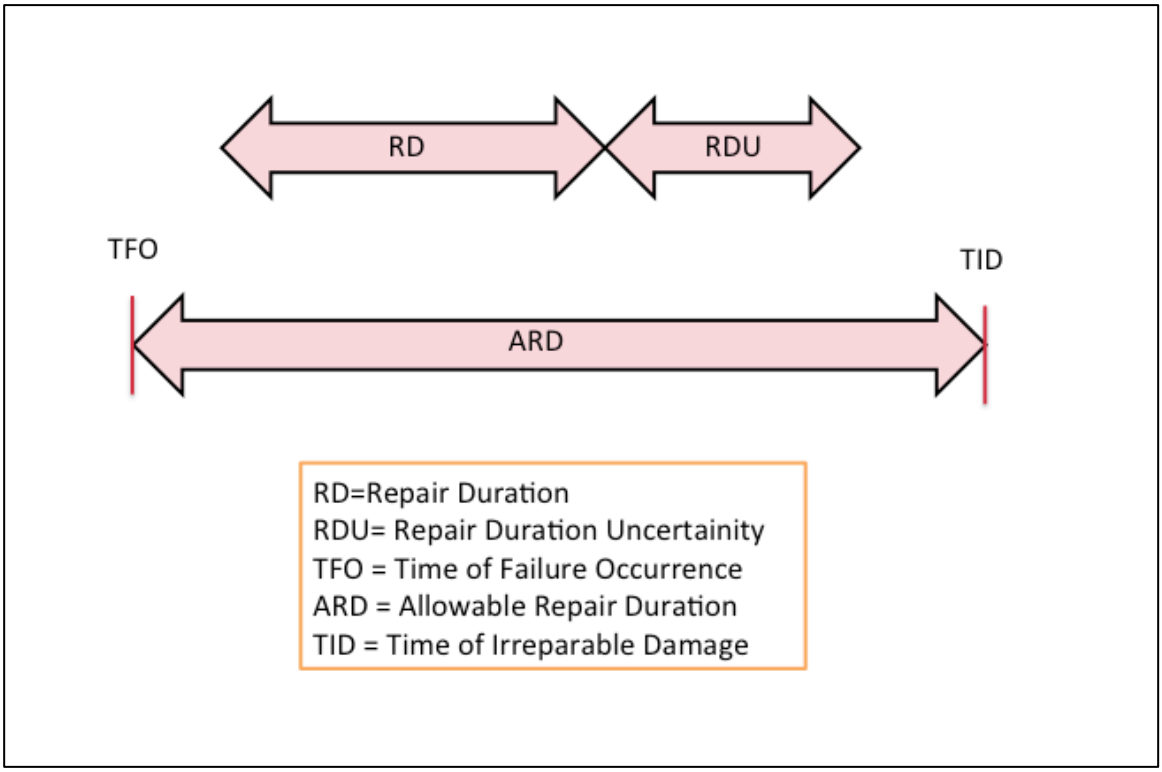

Figure 3. Failure timing elements required for the decision making process

For any particular component it is assumed that systems engineers can provide information on how long the repair duration (RD) and uncertainty of this number (RDU) would be. The Time of Failure Occurrence (TFO) is provided by the diagnostic system and associated telemetry information. Allowable Repair Duration (ARD) captures the notion that, for some components, unless repairs are done within a specified period of time, the component would be permanently damaged. For instance, a computer might not fail immediately with the failure of a cooling fan, but, unless the fan is fixed or replaced within a given period of time, the computer will be irreparably damaged.

Fault information within the fault table will also include pointers to explanations for the timing elements shown here. An important notion that was left to future development, for both activity and fault information, is that some elements will be editable by ground operators or crew. The idea is that a crew member, for instance, might decide that the crew will be able to complete a repair activity within a shorter time and with less uncertainty than indicated in the original table. This updated information would of course have an effect on the system's decision making process and advice provided.

V. Reasoner

The role of the Reasoner is to compute both proposed timing information for repair activities and to provide a value of Severity based on the number and criticality of the activities blocked by given faults and the functional failure of all impacted components. The timing calculation is simply algebraic and shown in Fig 4 where the latest Repair Start Time (RST) is computed by subtracting the Activity Duration, Repair Duration and its uncertainty from the Critical Effect Time, but it also needs to take into account the fact that the repair needs to be started before the Time of Irreparable Damage (TID), and that the restored activity needs to be performed before it becomes obsolete (TAE).

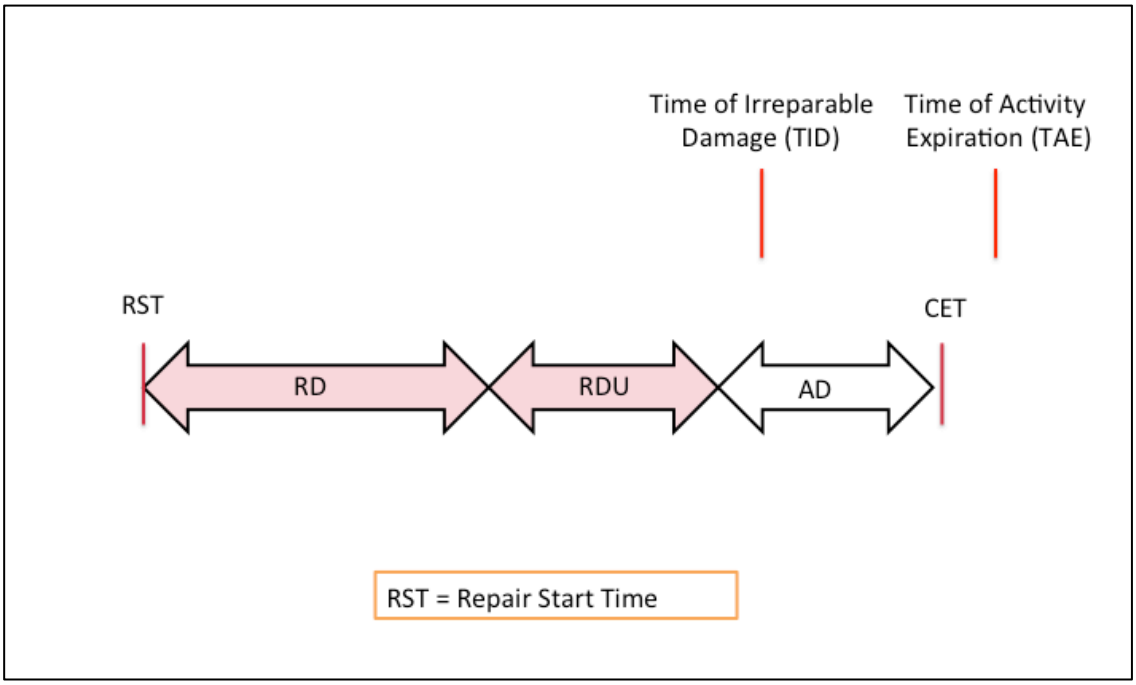

Figure 4. Basic computation of latest Repair Start Time

The computation of the Severity Value is more complex. We start by counting how many activities are blocked by each fault for each Criticality Value. In the example shown in Fig. 5, for Fault 1 we have 1 activity with CV =1, 2 activities with CV=2 and 5 activities with CV=5. For Fault 2 we have 0 activities with CV=1, 2 activities with CV=2 and 1 activity with CV=3. The figure also shows the histograms for the remaining faults – 3 and 4. Note that the criticality value range depends on how finely we want to distinguish the criticality of different activities; we find that 3 or 4 is typically sufficient. The number of faults and number of activities affected by them varies with the kind of mission that is being planned and system that is being modeled. The complete table is in Fig 6A.

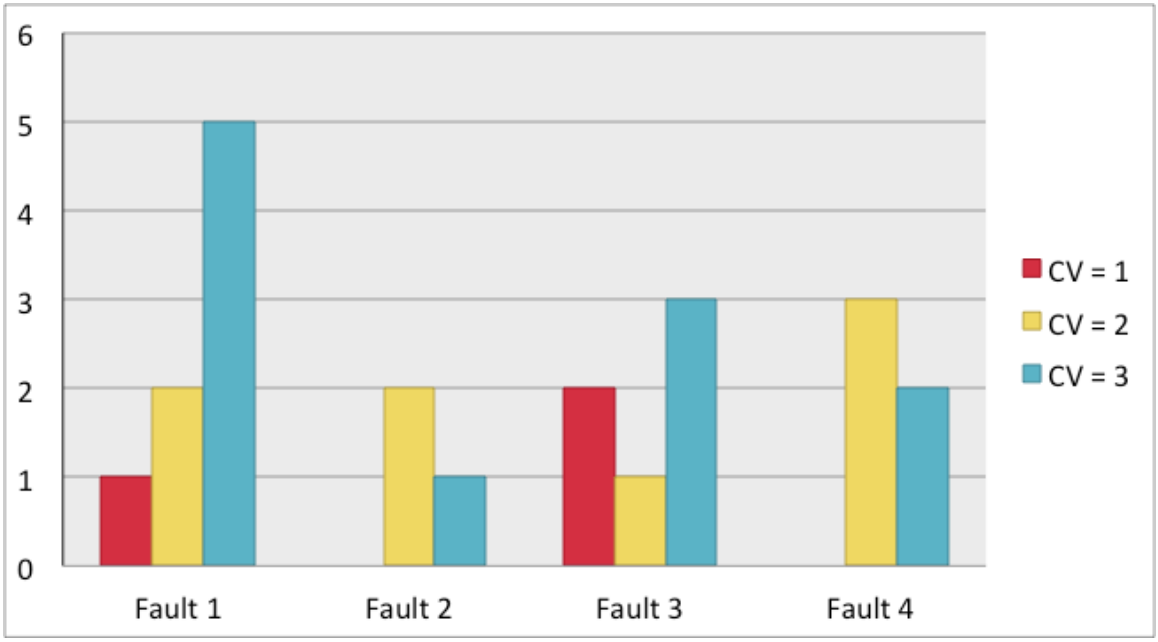

Figure 5. Criticality histogram for example with 3 Critcality Values and 4 faults.

The next step is to change the histogram numbers to percentages of the total number of activities that have a given CV in the system. For this example we assume that the total number of activities for each CV value is 5. For Fault 1, we find 5 activities with CV= 3. This is the total number of activities with CV = 3 within the system. The number "5" is thus converted to "100%". This needs to be approximated with "99" to avoid a "spillover" effect that will be explained below. All other numbers are also converted to percentages of "5". This is shown in Fig. 6B. Finally the percentages shown for each fault are concatenated and a Log is taken (Fig. 6B) to obtain a number that can provide an intuitive measurement of what we defined as "Fault Severity".

A. From the histogram:

	CV=1	CV=2	CV=3
F1	1	2	5
F2	0	2	1
F3	2	1	3
F4	0	3	2

B. Change to percentages, concatenate and take the log:

	CV=1	CV=2	CV=3	Concatenation	Log
F1	20	40	99	204099	5.3
F2	0	40	20	004020	3.6
F3	40	20	60	402060	5.6
F4	0	60	40	006040	3.8

Figure 6A,B. Steps for the computation of the Severity Value of faults

It should be clear now, why 100% is approximated with "99%". The extra digit would affect the order of magnitude of the Log function, or, if the "1" were added to the number in the CV=2 column it would imply that "100" activities of CV=3 are worth 1 activity of CV=2, and we would not want to be tied into such quantitative assumption in how the Criticality Values of different activities are selected. Strictly speaking, a final conversion to Log values is not necessary if we simply want to provide a ranking of faults, but we found it useful to come up with a simple meaningful number of the same order of magnitude as the Criticality Value even though the meanings are different. To reiterate, Criticality refers only to Activities and is NOT computed, rather it is assigned *a priori* on the basis of knowledge of the mission and its systems, whereas Severity refers only to Faults and is computed on the basis of the impact of the Faults on Activities. It is not possible to know the effect of any particular failure until we know what activities are affected by the lack of resources due to the failure. The failure of a power supply has very different severity if its effect is to prevent a secondary scientific experiment to be performed as opposed to preventing the deployment of a re-entry parachute.

VI. User Interface

The user interface for the FRAd will be integrated in the framework of the ACAWS interface. In the ACAWS GUI the entire system can be represented in movable panes to show fault locations and fault consequences (impacts) (Fig. 7). The FRAd GUI is still in prototype form and is designed mainly with crew interaction in mind, although of course the information provided could facilitate decision support on the ground, and potentially be useful in ground operator training.

The fundamental purpose of the FRAd system is shown in the three prototype screen shots shown below. The model system used for the prototype demo was a simplified (non-ITAR) model of the EFT-1 capsule. The model had about 1000 components and 1800 faults (failure modes). For FRAd development, we created a representative mission schedule with 27 activities and focused on 30 possible faults. All fault and activity information was stored in the Fault Table and Activity Information Repository, respectively (Fig.

1), but, for the purpose of our prototype that information was not validated as in the context of a mission. Criticality values and timing information were constructed simply for the purpose of verifying the computation of severity values and repair start times. With this proviso in mind the screen shots show the following example. Three faults are identified in the system and their Severity Value is computed, together with latest start times for repair and time remaining from current time (Fig. 8). The crew is made aware of the high Severity Value for two of the faults and how much time is left for action. The highest severity is 7.9, but repair can wait 3 days for that problem. The crew decides to look first at the fault with severity 7.5 (Fig. 9). The pane that details this fault shows the list of activities that are blocked as a consequence of this failure. One of these activities has Criticality Value of 1, which typically indicates life and/or mission critical events. The presence of this high criticality activity explains the high severity value of the fault. The crew can then drill deeper into this activity (Fig. 10) to find when it is scheduled to happen, why it is critical, what other resources are required for that activity (besides the one that failed).

Together these elements form the basis of the decision making process for what needs to be done, when and why.

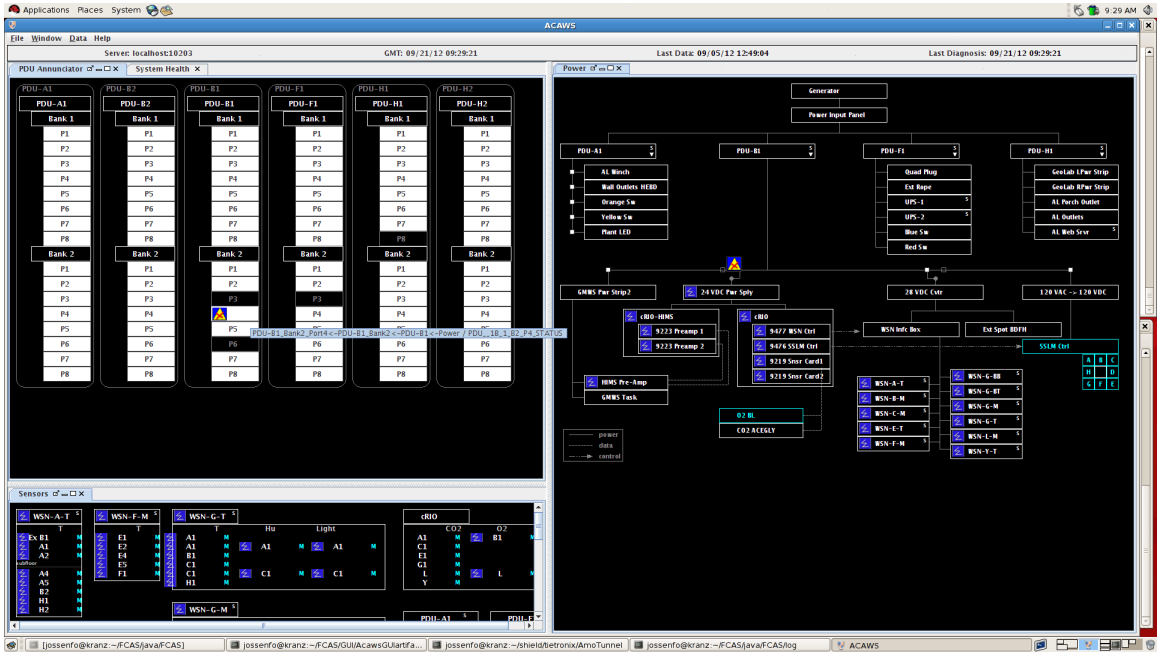

Figure 7. The ACAWS GUI allows for different levels and types of information arranged in independent window panes according to the individual requirements of operators. The whole system can be displayed at different hierarchical levels and icons "light up" to show component failures (yellow triangle) and impacted components (blue icons).

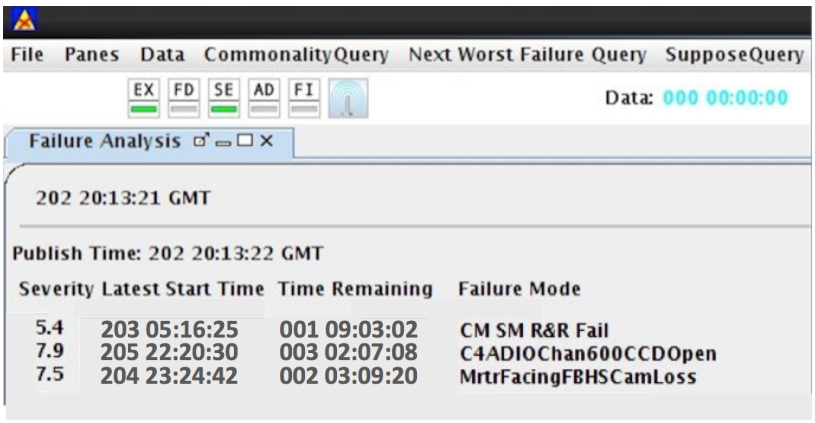

Figure 8. Three faults are identified by the system and are displayed together with their Severity Values, Latest Start Time for repair procedures and Time Remaining. The first three-digit number is "day of the year", then the time is displayed in hh:mm:ss.

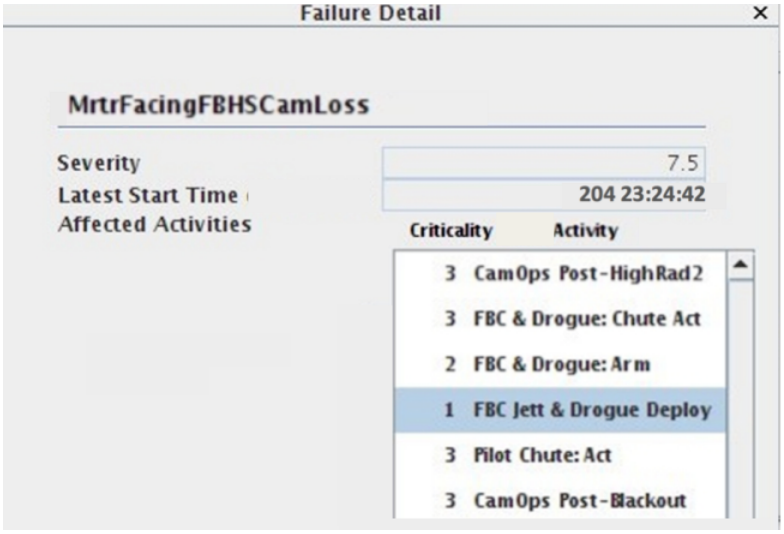

Figure 9. Detail of the failure shows the list of activities that are blocked by the failure, together with their Criticality Value (CV). Further information can be retrieved for each activity.

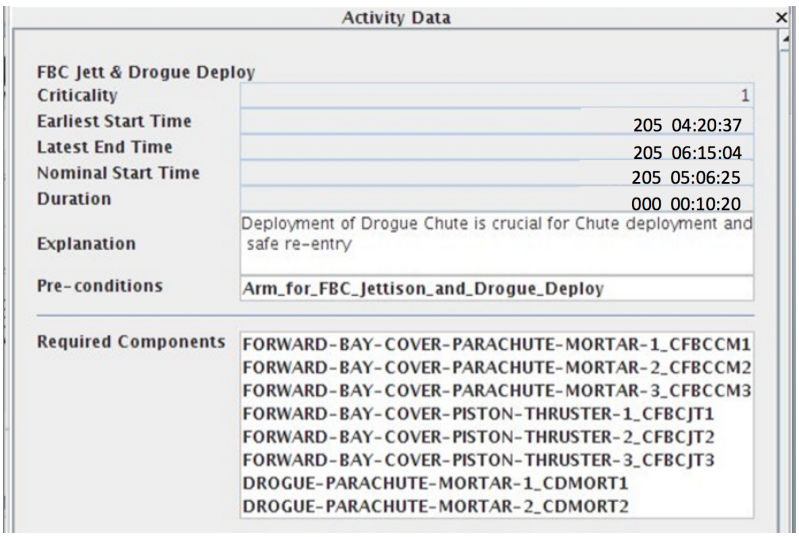

Figure 10. The activity with Criticality Value = 1 is further examined to see when it is expected to take place (if the fault is repaired), why it has CV = 1 and what other resources are required for the activity to be performed.

VII. Conclusion

Crewed space travel, even as far as the Moon, has been relying on ground support for any kind of decision making regarding systems failures and timing of repair procedures. For deep space travel to the vicinity of Mars and asteroids, communication latencies can be over an hour and the crew may need to be informed more quickly about the gravity of any system failure and about the necessary timing of repair procedures. For this purpose we have extended the capabilities of ACAWS, our diagnostic and failure consequences analysis system, to provide information on the Severity of failures and timing for initiation of repair procedures. The concept of Severity is quantitative, based on the number and Criticality of activities blocked by the failure. For future implementation we foresee an additional pass of the planning system that may need to sort out potential conflicts in needed repair procedures and new resource requirements, in light of the failure that have been identified, but, even just with this first step, we believe that crew action would be guided by clear quantitative information for safer and quicker decision making. The system is also envisioned for use as an operator training tool, where hypothetical faults could be presented and the operator's decisions could be compared with those of the system. Clearly this could become part of the rigorous validation process that would be required for appropriate system accreditation.

VIII. References

1. Lilly Spirkovska, Gordon Aaseng, David Iverson, Robert S. McCann, Peter Robinson, Gary Dittemore, Sotirios Liolios, Vijay Baskaran, Jeremy Johnson, Charles Lee, John Ossenfort, Mike Dalal, Chuck Fry, Larry Garner, "Advanced Caution and Warning System, Final Report – 2011," NASA/TM-2013-216510, March 2013.

2. Silvano Colombano, Vijayakumar Baskaran, Lilly Spirkovska, Gordon Aaseng, Irene Smith, Mark Schwabacher, John Ossenfort, and Robert S. McCann, "Fault Consequence Analysis for Space Missions," AIAA SPACE 2013 Conference & Exposition, San Diego, CA, September 10-12, 2013.

3. J.J. Marquez, M. Ludowise, M. McCurdy, M., and J. Li, "Evolving from Planning and Scheduling to Real-Time Operations Support: Design Challenges," In Proceedings of 40th International Conference on

Environmental Systems. Barcelona, Spain, 2010.

4. Paul Morris, Minh Do, Robert McCann, Lilly Spirkovska, Mark Schwabacher, and Jeremy Frank, “Integrating System Health Management and Planning & Scheduling to Determine & Recover from System Failure Effects”, Systems Demonstrations and Exhibits program, 24th International Conference on Automated Planning and Scheduling (ICAPS 2014), Portsmouth, NH, June 21-26, 2014.