Determining Mission Effects of Equipment Failures

Paul Morris[1] and Minh Do*[2] and Robert McCann[3] and Lilly Spirkovska[4] and Mark Schwabacher[5] and Jeremy Frank[6]
NASA Ames Research Center, Moffett Field, CA 94035

**Stinger Ghaffarian Technologies, Inc., NASA Ames Research Center, Moffett Field, CA 94035*

NASA plans call for long duration deep space missions with human crews. Because of light-time delay and other considerations, increased autonomy is needed. Crews on next-generation missions will likely be small, perhaps with as few as four members. A small crew is not likely to possess the full range of expertise needed to deal with unexpected failures and anomalies. Applied artificial intelligence technologies have developed decision support tools with the potential to fill the gap, but these tools need to be integrated to provide a smooth operational capability. In this paper we describe such an integration involving anomaly detection, diagnosis, system effect propagation, and plan repair.

Nomenclature

ISHM = Integrated Systems Health Management
P&S = Planning and Scheduling
AS = Autonomous Systems
ACAWS = Advanced Caution and Warning System
C&W = Caution and Warning
IMS = Inductive Monitoring System
EUROPA = Extendable Uniform Remote Operations Planning Architecture
SPIFe = Scheduling and Planning Interface for Exploration
TEAMS RDS = Testability, Engineering, and Maintenance System Remote Diagnostic Server
FMECA = Failure Modes, Effects, and Criticality Analysis
DSH = Deep Space Habitat
MOT = Mission Operations Test
STN = Simple Temporal Network
AD = Activity Dictionary
MAPGEN = Mixed Initiative Activity Plan Generation
LADEE = Lunar Atmospheric Dust Environment Explorer
NDDL = New Domain Description Language

I. Introduction

Future NASA plans call for long duration deep space missions with human crews. Because of light-time delay and other considerations, it will not be feasible to micro-manage the mission from the ground to the extent that was done for the Apollo missions to the Moon; thus, increased autonomy is needed. Managing missions with less real-time ground assistance poses a number of challenges, however.

In current practice, detection, isolation, and recovery from spacecraft system faults requires large teams of specialized ground-based controllers that must mentally integrate data gathered via disparate operations products. Failure management is even more difficult, and more reliant on ground-based expertise, for multiple failures or unexpected anomalies. Crews on next-generation missions will likely be small, perhaps with as few as four

[1] Code TI, M/S 269-1, NASA Ames Research Center.
[2] Stinger Ghaffarian Technologies Inc., NASA Ames Research Center.
[3] Code TH, M/S 262-4, NASA Ames Research Center.
[4] Code TI, M/S 269-3, NASA Ames Research Center.
[5] Now at Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA.
[6] Code TI, M/S 269-3, NASA Ames Research Center.

members. Even with extensive training, a small crew is not likely to possess the full range of expertise needed to deal with unexpected failures and anomalies.

Fortunately, applied artificial intelligence technologies associated with the emerging fields of Integrated Systems Health Management (ISHM) and Planning and Scheduling (P&S) have matured to the point where they can begin to provide advanced decision support products for crew-members and flight controllers alike. NASA's Autonomous Systems (AS) project is exploring the use of advanced automation tools to support such tasks as planning, plan execution, anomaly detection, diagnosis, and failure recovery. Several such tools have been developed in past years. In conjunction with building generic tools, much of the work involves developing system models that capture system structure, interconnections, tests, procedures, and failures. System models can support a wide variety of tools and fault management capabilities.

A recent focus of our work has been on integrating these tools so that they work together to provide a smooth operational capability. In the past year, we have addressed the integration of three key tools, an Advanced Caution and Warning System (ACAWS), the Scheduling and Planning Interface for Exploration (SPIFe), and the Extendable Uniform Remote Operations Planning Architecture (EUROPA). The objective of our tool integration effort is to assist controllers and crew with determining effects of system failures on mission objectives and re-planning the mission activities timeline to recover from failures and accomplish as many preplanned mission activities as possible. Thus, the integrated system provides a capability for detecting and diagnosing faults, analyzing the disabling effects of the faults on downstream components, determining the impact of those disabling effects on the existing crew activity schedule, and revising the schedule to cope with the faults.

To achieve this capability, we have developed methods to communicate failures between ISHM and the planning tools, compactly specify resources required by timeline activities, customize interfaces to facilitate real-time interaction, and utilize and extend the ACAWS, SPIFe, and EUROPA tools.

In the following sections, we describe the component tools, the roles they play in the combined system, the information flow between the tools, and the integration "glue" used to link them together. We also discuss utility tools used to facilitate model building for the P&S components.

II. Decision Support Tools for Spacecraft Mission Management

In this section we describe the ACAWS (anomaly detection and diagnosis), SPIFe (plan visualization and editing), and EUROPA (plan generation) tools that are the components of the integrated system.

A. ACAWS

The Advanced Caution and Warning System (ACAWS) supports spacecraft failure management for human missions to distant destinations and human missions in low Earth orbit with smaller, less experienced ground controller teams. Previous manned spacecraft have relied on a caution and warning (C&W) system primarily implemented as threshold-based alarms issued by individual components and subsystems. A single fault can produce a multitude of C&Ws as each component affected by that fault issues its own alarm(s). Flight controllers train for years to develop necessary expertise for a specific discipline (subsystem) and work as a multi-discipline team to make sense of these numerous, nearly simultaneous C&W messages to deduce the instigating "root cause" fault(s), determine the effects of the fault(s) on the spacecraft and mission, and adapt workaround procedures to safely and effectively proceed with the mission. As humans venture farther from Earth, a paradigm change is required from current operations because the crew will not be able to depend on support from ground as readily as they do now when communication speed—limited by the speed of light—is noticeably slower (up to 22 min one-way to/from Mars) or when knowledge becomes more limited within the on-console ground teams as budget cuts and retirements force smaller, less experienced teams.

ACAWS provides an integrated toolset that includes anomaly and fault detection, automated diagnostic analysis and root cause identification, troubleshooting assistance when instrumentation alone is insufficient to provide an unambiguous diagnosis, effects assessment, and a dynamic and interactive graphical representation of the spacecraft systems and their health.

For anomaly detection, ACAWS utilizes the Inductive Monitoring System (IMS) developed at NASA Ames Research Center (ARC) (Iverson et al. 2012).[1] IMS uses clustering to learn nominal system behavior from archived or simulated system data, automatically builds a "model" of nominal operations, and stores it in a knowledge base. The IMS real-time monitor and display informs users of degree of deviation from nominal performance. Analysis can detect conditions that may indicate an incipient failure or required system maintenance.

For automated diagnosis and troubleshooting, ACAWS utilizes the Qualtech Systems Inc. TEAMS RDS (Testability Engineering and Maintenance System Remote Diagnostic Server) product (http://teamsqi.com). TEAMS

is a model-based system enabling a general diagnostic reasoner to be applied to any physical system by plugging in a failure propagation model of the system and real-time observations about the system's behavior. This model captures considerable system knowledge gathered from FMECA reports, fault trees, schematics, instrumentation lists, operational use cases, other technical documentation, and system engineering expertise. For real-time use, the model is reduced to the relationship between various system failure modes and system instrumentation. Instrumentation readings are received, processed, and turned into pass/fail results for specific tests. Abductive reasoning is then used to determine the set of failure modes that explain the test result signature.

In addition to providing knowledge necessary for diagnosis, the diagnostic model captures knowledge necessary for system effects reasoning. However, unless the model is developed with system effects in mind, diagnostic models generally encode only 80%-90% of effects of a failure. For the exceptional 10%-20% of effects that cannot be determined from a diagnostic model alone, targeted knowledge can be added to the model in the form of Boolean expressions that indicate whether a component is impacted by a failure or not, and whether an impact should propagate further in the model. The ACAWS System Effects reasoner (Columbano et al., 2013),[2] developed at NASA Ames, exploits these augmented models to determine the full set of effects of a failure on the spacecraft (or generally, on the system of interest).

The final module of ACAWS is the operator interface. The operator is provided the results of each of the reasoners – anomaly detection, diagnosis, and system effects – in multiple formats designed with the operators' tasks in mind. Text log based formats provide all the details about either diagnosis or effects in one place that can be copied from ACAWS into other documents for communication purposes (say, between crew and ground, or in an anomaly resolution report). Component based formats, together with the component hierarchy extracted from the diagnostic model, provide annunciator panel type displays that summarize the spacecraft's health. Components can also be arranged to reflect the architecture of subsystems, providing a visual indication of the flow of a failure and its effects. Symbols with supplementary color-coding added to the components provide visual indication of failure; possible failure (that is, a component that is a part of an ambiguous diagnosis); or affected by a failure or possible failure, either total loss of function or loss of redundancy. In addition to displaying real-time information, the ACAWS operator interface provides task-facilitating query options. For example, the operator can suppose component(s) failed or may have failed and query the system for the effects of that scenario – very useful for operator training or to explore hypothetical "what-if" situations while monitoring a flight. Also, the operator can ask for common ancestors of selected components, useful for determining next-worst failure – an analysis often at the forefront of a controller's failure management process, or can ask for the path from the failure(s) to a selected component to analyze the propagation from, say, an electrical power system failure to a coolant pump. Figure 1 shows one view of an operator-customizable arrangement of information panes in a hybrid real-time plus analysis mode (McCann et al., 2013).[3]

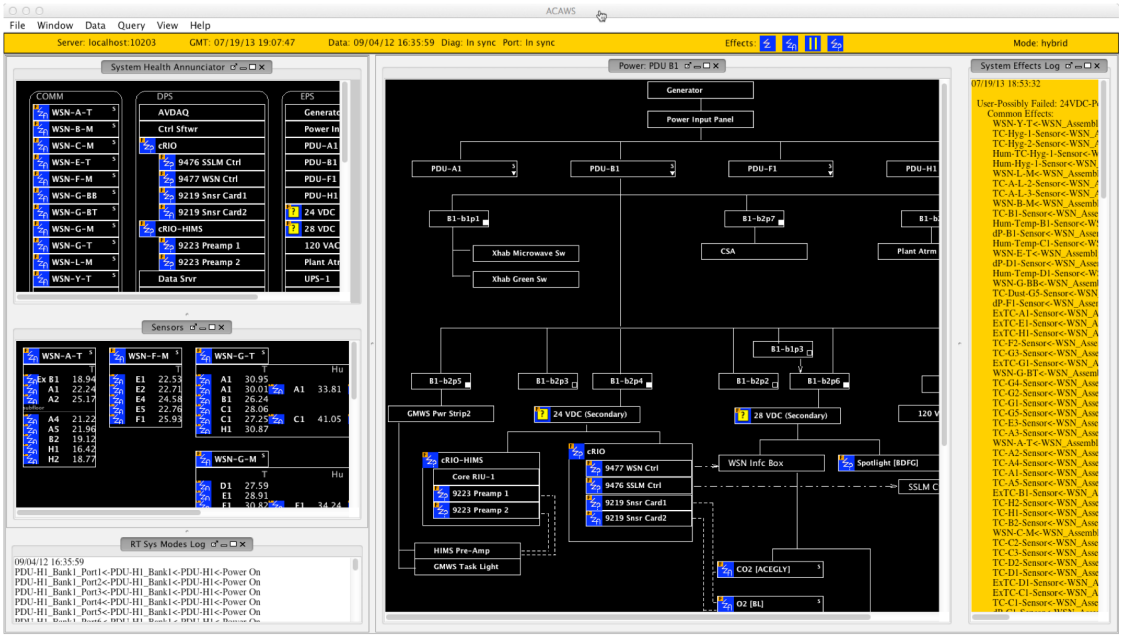

Figure 1: ACAWS Information Panes.

ACAWS has been evaluated and matured through a variety of Earth-analog mission operations tests, such as the Habitat Demonstration Unit Desert Research and Technology Studies (D-RaTS: www.nasa.gov/exploration/analogs/desertrats), Deep Space Habitat (DSH) Autonomous Mission Operations evaluation (Frank et al., 2013),[4] and the Deep Space Habitat Mission Operations Test. Most recently, it is being applied to a ground-based capability demonstration during Exploration Flight Test 1 (EFT-1) – an unmanned test flight of the next-generation manned spacecraft – Orion – being designed for missions to an asteroid and eventually to Mars.

B. SPIFe

SPIFe (Marquez et al., 2010)[5] is used for plan creation, validation and editing. It was developed as a joint project at NASA Ames and NASA's Jet Propulsion Laboratory (JPL). SPIFe was built using the Eclipse development environment, which facilitates customization using separate plugins.

The tool includes a sophisticated plan database and graphical user interface to detect flaws in the plan that can be interactively fixed by the user. These include situations where an activity violates user-specified temporal constraints, over-subscribes resources, or fails to meet state requirements. The tool specifies the type, time of occurrence, and culprit activities for each flaw. The user then has the option of adding, deleting, or moving activities, or modifying activity parameters, in order to resolve each flaw. Flaws may also be waived by the user.

The static properties of activities are specified by means of an *Activity Dictionary* (AD), which is a type of model of the application domain. In the AD one can define state variables, as well as resource types and resource limits, and then specify the state requirements and effects of activities, and their impact on resources. The AD is procedurally oriented, allowing the use of javascript to calculate resource usage and other values, based on activity parameters and other variables.

SPIFe is designed for interactive use with an extensive graphical user interface (GUI). The GUI provides several panes, or *views* of the plan, for different types of user interaction. The Timeline view presents a Gannt-like picture of the activity schedule with the activities arranged chronologically from left to right, and organized into separate rows according to user-selected options such as by category or parameter value. It can also be toggled to show instead a tabular listing of the activities. The Plan Advisor view shows detailed information about the flaws, which are also flagged in the Timeline view. An Activity Dictionary view can be used to drag new activities into the plan, while a Details view focuses on individual activities and their parameters. The different views are fully integrated, and selections and changes in each are reflected in the others. There is also a Plan Navigator pane for selecting and opening previously saved or imported plans. Several plans may be open at the same time, using different tabs, but only the selected tab is visible in the views.

The SPIFe tool was used for ground planning on the Mars Phoenix mission and is in current use for the Mars Science Laboratory Curiosity rover as well as various applications for the International Space Station. It has also seen use in several analog operations tests and studies.

C. EUROPA

Developed at NASA Ames, EUROPA (Frank and Jonsson, 2003),[6] supports automatic planning with temporal, state, and numeric constraints. In contrast to the procedural orientation of SPIFe, it is based on a declarative language similar to those used in the automated planning community, and utilizes a least commitment planner. An early version of EUROPA was incorporated into the MAPGEN tool used in tactical planning for the Opportunity rover in the continuing Mars Exploration Rover mission. It also traces its heritage to the automatic planning component of the Remote Agent Experiment, where an onboard artificial intelligence (AI) system controlled the Deep Space I robotic spacecraft for several days (Muscettola et al., 1998).[7]

The EUROPA database provides a plan consisting of a set of activities that are interrelated via a rich set of temporal, state, and resource properties. In particular, it includes a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl, 1991)[8] to support planning and ensure consistency of the temporal constraints. The STN determines a plan that has *temporal flexibility*; that is, it corresponds to a set of related schedules rather than a single schedule. Instead of having a single time, the events in the plan have a *time interval* with lower and upper bounds on when they can occur. This provides scope for adjusting to temporal deviations during execution.

The current tool has been integrated with SPIFe using a specialized EUROPA plugin, called DynamicEuropa. This combination provides for a Mixed-Initiative Planning framework (Bresina and Morris, 2007),[9] in which a human operator collaborates with the Europa suite of automated tools to create a plan. In the integration, SPIFe presents to the user a single *nominal* schedule, which is backed by a flexible plan in EUROPA. When adjusting the plan to satisfy constraints, the flexible plan is restricted to exclude constraint violations. EUROPA then computes a new nominal schedule that satisfies the constraints while minimizing the changes from the previous one, and communicates it to SPIFe. This approach serves to maintain general plan stability and allows the human operator to

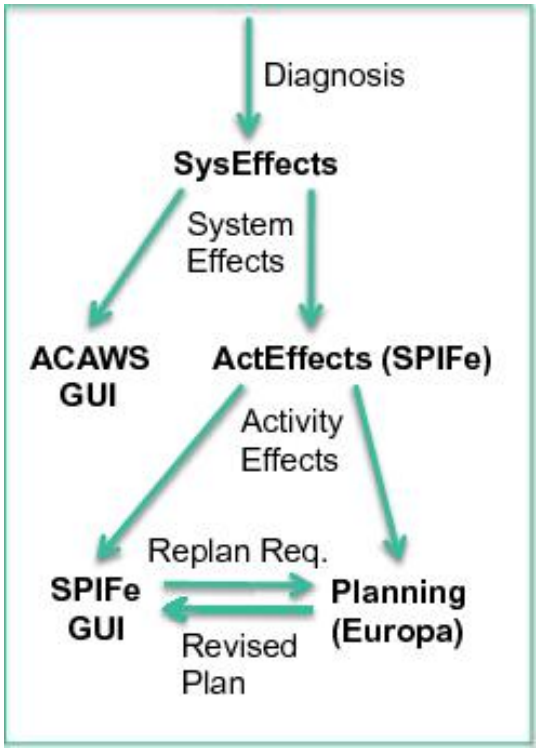

Figure 2: System Architecture.

express simple timing preferences. The nominal schedule can also provide heuristic guidance to an automated planner (Morris et al., 2011)[10] through the ordering of events.

This general approach was used for NASA's recent Lunar Atmospheric Dust Environment Explorer (LADEE) mission. However, the automatic planning capability of DynamicEuropa in LADEE and prior SPIFe integrations is limited to removing mutual exclusion flaws. For the AS project we have implemented a more general automatic solving capability that reschedules activities based on state requirements and effects. For example, the LADEE planner can automatically move activities away from forbidden regions but it does not automatically move them into required regions. The new solver can do so, and can also move activities that supply missing effects from an activity reserve area to where they are needed in the plan.

III. Integration

Figure 2 shows the overall architecture and information flow of the integrated system. The modules communicate with each other through publish-and-subscribe middleware. The System Health module, which includes TEAMS and ACAWS, provides a diagnosis of the failure and determines its downstream consequences. The *System Effects* are those components that are disabled (no longer functioning as designed) as a result of the components that have failed. (For example, a lamp may be disabled because the module that supplies power to the lamp is broken.) *System Effects* are computed using a propagation approach that traverses the causal chain of physical functionality between the components. The ACAWS software reuses the diagnostic model built for use with TEAMS but augments it with effect propagation logic where necessary.

The Planning module, composed of SPIFe and EUROPA, determines the *Activity Effects* on the mission timeline, that is, activities that can no longer occur because they depend upon resources that have either been declared failed or non-operational by ACAWS. For example, suppose an astronaut was scheduled to perform a visual examination of a mineral sample using light from the lamp to which we referred earlier. Obviously, the examination could no longer be performed (the activity could not be carried out) if the lamp was unable to provide illumination (as would occur if a failure in the electrical power system cut off power to the lamp). In that case, the plan needs to be modified to either fix the failure, thereby restoring power to the lamp and allowing the scheduled activity (sample inspection) to proceed, and/or to meet as many mission objectives as possible if the failed component could not be fixed. Replanning of this sort takes advantage of the mixed-initiative capability of the SPIFe/EUROPA combination. For example, the operator can make manual changes, or can call on the automatic planning capability

Figure 3: Plan with Flaws.

in EUROPA to assist. In our experiments, EUROPA automatically added repair activities to fix broken electrical power system components and adjusted the crew schedules accordingly.

In previous applications, such as daily tactical planning for Mars robotic missions, SPIFe has been used in a standalone mode. For our purposes, a more interactive mode was required, in which SPIFe "listens" to the failure diagnoses coming from ACAWS and automatically updates the plan in a way that shows the effects on the activities. To achieve this, we modified a feature, called the *InCon,* of the standalone tool. The InCon is used to specify initial conditions, which may include initial resource levels and initial states, such as the broken or disabled states of the components. Normally, the InCon is loaded by manually importing a file. In our case, we modified the code to periodically look for a new version of the file and use it to overwrite the existing initial conditions. We also implemented a small process that subscribes to the failure reports and writes a corresponding InCon file.

The SPIFe model involves an internal Activity Dictionary that has a complicated format. While it is possible to load an xml-based text version of the AD, it is a time-consuming task to author such a file for a significant application, especially with the frequent changes that occur during development. To facilitate this work, we developed a general high-level language for specifying the model, and wrote an automatic translator for converting it to the AD format. For this particular application, we also accepted as input a spreadsheet method of specifying the activity types and their properties. This provided a more comfortable interface for our domain expert. In previous integrations, a separate translator is needed to automatically extract the NDDL model needed by Europa from the SPIFe AD. For our application, this required significant modification to support the capability to introduce new state-changing activities and move them to where they are needed in the plan.

The Activity Effects correspond to flaws in the SPIFe Timeline and Plan Advisor views. In the Timeline view, these show up as red tick marks at the top and the culprit activities are depicted with red borders (see Fig. 3). Additional information is available by mouse-over. The Plan Advisor gives even more details and also provides a fixViolations button, which calls Europa to do automatic replanning. In our application, replanning placed repair activities at appropriate points in the plan, and rescheduled other activities around the repair so that crew-members were not double-booked.

When the call to fixViolations returns, it presents a set of recommendations in the form of a table that lists activities with their old and new start times. The user may accept or reject any or all of these. The accepted changes are installed in the plan. Conveniently, SPIFe provides an undo/redo feature for all modifications to the plan, including fixViolations, which allows toggling between the old and new plans.

IV. Tests

We constructed nine scenarios to test the integrated system. This involved selected activities from the Deep Space Habitat Mission Operations Test timeline for four crew-members, the Commander (CDR), and three Flight Engineers (FE1, FE2, FE3). The activities were associated (Fig. 4) with pieces of equipment (resources) that depended on components of the avionics power system, so failure of the components impacted crew activities. All but one of the scenarios involved an individual fault which varied according to its level in the electrical power system hierarchy (and hence, the ACAWS diagnostic hierarchy). Thus, a high-level fault would typically have a large impact on the activity schedule, affecting several individual activities, while a low-level fault might have little or no impact. One scenario involved two independent faults.

The faults were injected into ACAWS using a "What-If" testing capability. The resulting system effects were transmitted to SPIFe where the component failures were interpreted as unsatisfied state requirements. In each case, EUROPA was then called to fix the violations. One fault scenario did not impact any activities. For the eight other scenarios, solutions were found without backtracking in all cases, with no domain-specific or complex heuristics. The nominal (original) plan had 36 activities. A few activities required two crew members. Automatic solving took about a second on a Mac laptop, which allowed for a comfortable interaction with ACAWS.

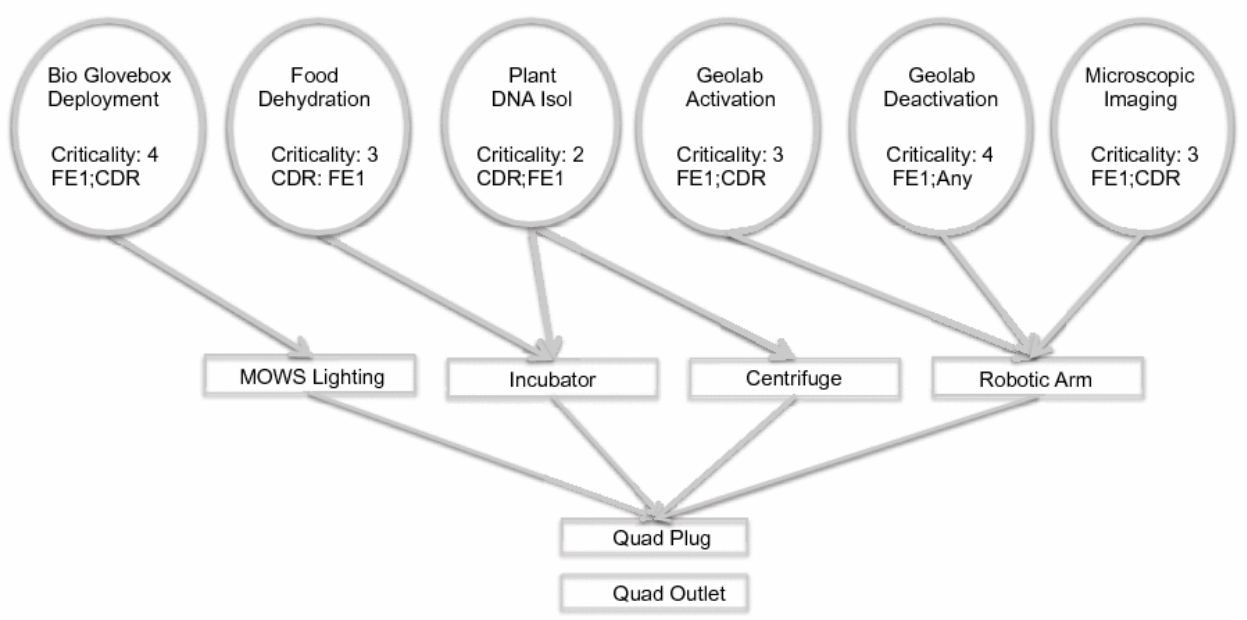

Figure 4: Association of Activities to Resources.

V. Closing Remarks

Apart from extensive changes that were needed to support enhanced flaw removal in the fixViolations functionality, the integration mainly involved modification of the interfaces. We also changed some look-and-feel aspects of the GUI to better suit the application.

In the scenarios considered, the plans to be repaired were assumed to happen after the time that the resources became broken and disabled. In a more continuous or ongoing planning/execution scenario, an unfolding plan would need to be truncated to future activities before the repair process is applied. Thus, activities in the past would be discarded and activities that are partially executed would be shortened. In that case, the resource state modifications might need to be applied to the measured or projected conditions at the current time rather than the original initial conditions. The SPIFe tool does include a limited capability to project future conditions, but that might require extensions for such scenarios.

Search control was relatively straightforward for the type of planning needed for the tests. In the future, we would like to address more challenging planning tasks such as synthesizing novel procedures from basic "atomic" actions. Also needed are mechanisms to automatically generate planner models from more neutral knowledge sources that could be shared by many tools.

Acknowledgments

The authors wish to acknowledge the contributions of the ACAWS and SPIFe/EUROPA development teams, including Gordon Aaseng, Vijay Baskaran, Adam Campbell, Silvano Colombano, Mike Dalal, Chuck Fry, David Iverson, John Ossenfort, Brandon Oubre, Irene Smith, Arash Aghevli, Javier Barreiro, John Bresina, Steve Hillenius, Bob Kanefsky, and Jessica Marquez. This research was supported by NASA's Space Technology Mission Directorate through the Game Changing Development Program's Autonomous Systems Project.

References

[1]Iverson, D.L., Martin, R., Schwabacher, M., Spirkovska, L., Taylor, W., Mackey, R., Castle, J.P., and Baskaran, V., "General Purpose Data-Driven System Monitoring for Space Operations." *Journal of Aerospace Computing, Information, and Communication* 9:2, 2012.

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

[3]Frank J., Aeseng G., Dalal K.M., Fry C., Lee C., McCann R., Narasimhan S., Spirkovska L., Swanson K., Wang L., Molin A., and Garner L, "Integrating Planning, Execution, and Diagnosis to Enable Autonomous Mission Operations," *Scheduling and Planning Applications Workshop (SPARK)*, Rome, Italy, June 10-14. 2013.

[4]McCann R.S., Spirkovska L., and Smith I.. "Putting ISHM Capabilities to Work: Development of an Advanced Caution and Warning System for Crewed Spacecraft," *AIAA Modeling and Simulation Technologies (MCT) Conference,* Boston, MA, August 19-22, 2013.

[5]Marquez, J. J., Ludowise, M., McCurdy, M., and Li, J.. "Evolving from Planning and Scheduling to Real-Time Operations Support: Design Challenges." *40th International Conference on Environmental Systems*. Barcelona, Spain, 2010.

[6]Frank, J. and Jonsson, A.. "Constraint-Based Interval and Attribute Planning." *Journal of Constraints* 8:4 Special Issue on Constraints and Planning, 2003.

[7]Muscettola, Nyak, Pell, and Williams. "Remote Agent: to Boldly Go Where No AI System Has Gone Before." *Artificial Intelligence* 103 (1-2), pp 5-47, Aug. 1998.

[8]Dechter, R., Meiri, I., and Pearl, J.. "Temporal Constraint Networks." *Artificial Intelligence* 49:1-3, 1991.

[9]Bresina, J., and Morris, P.. "Mixed-Initiative Planning in Space Mission Operations." *AI Magazine* 20:2, 2007.

[10]Morris P., Bresina J., Barreiro J., Iatauro, M., and Smith T.. "State-Based Scheduling via Active Resource Solving." *4th IEEE International Conference on Space Mission Challenges for Information Systems (SMC-IT*), 2011.