A Failure Recovery Planning Prototype for Space Station *Freedom*

David G. Hammen and Christine M. Kelly

The MITRE Corporation
1120 NASA Road One
Houston, Texas 77058

Abstract

NASA is investigating the use of advanced automation to enhance crew productivity for Space Station *Freedom* in numerous areas, including failure management. This paper describes a prototype that uses various advanced automation techniques to generate courses of action whose intents are to recover from a diagnosed failure, and to do so within the constraints levied by the failure and by *Freedom's* configuration and operating conditions.

Keywords: Planning, Failure Management, Artificial Intelligence

Introduction

Artificial Intelligence (AI) has been successfully applied to the fields of failure diagnosis and plan generation. This paper addresses the application of AI planning techniques to failure recovery. The Recovery expert (Rx) is an existing prototype that generates plans whose intents are to recover from diagnosed failures on the Space Station *Freedom* and to do so within the constraints levied by the failure and by other current operating conditions. The Rx uses artificial intelligence techniques (such as pattern matching, symbolic reasoning, and goal-directed planning) and statistical techniques (multiple criteria decision making) to generate the courses of action.

Background

The National Aeronautics and Space Administration (NASA) is in the process of designing and implementing a permanently manned space station, Space Station *Freedom*, to be put into low earth orbit. *Freedom* has a hierarchical, distributed command and control architecture. The highest level in the architecture is concerned with global, vehicle-wide issues and includes the people controlling *Freedom* and the software that they use to do their jobs. Our failure management prototype, initially described in (Hammen, Baker, Kelly, and Marsh, 1990) and expanded upon in this report, demonstrates the applicability of expert system software to provide decision support to *Freedom's* crew in recovering from vehicle failures.

This prototype is implemented in ART/Ada™, TAE™ Plus, and Ada, and comprises the Diagnostic Reasoner (DR), which diagnoses and analyzes the failure, and the Rx, which formulates courses of actions to take for recovery (Baker, Hammen, Kelly, and Marsh, 1991). This report focuses on the Rx; a companion article (Baker and Marsh, 1991) focuses on the DR. The DR and Rx are not completely automated functions: they are decision aids, and the user is provided with interaction capabilities throughout the failure management process.

The key technologies used in the implementation are model-based reasoning (for the DR) and goal-directed planning (for the Rx). The objective of the prototype is to show how we use these technologies and how the technologies work together synergistically. By demonstrating the value of this approach to failure management we hope to influence how *Freedom* failure management software will eventually be implemented.

Figure 1 shows an overview of the environment in which the DR and Rx prototypes work. The DR receives status and configuration reports that reflect changes in the operational components. In the case of a failure, the DR diagnoses the failure and the Rx formulates a set of possible solutions to the failure. Once a specific Course of Action has been selected, another prototype not described here, the Procedures Interpreter, sends commands that implement the selected Course of Action.

Failure Management Terminology

A fault is a "known or hypothesized defect in hardware or software causing error" (MDSSC, 1989). A fault may manifest itself as a failure, which is "the inability of a system, subsystem, component, or part to perform its required function" (MDSSC, 1989).

DR Overview (Baker and Marsh, 1991)

When the DR receives reports that indicate a failure it looks for possible causes, or suspects. More than one possible cause might be proposed for a given set of symptoms, and more than one independent fault might be proposed as a possible cause. The DR projects the effects of the possible causes into the future, producing failure modes and effects criticality analyses, or Impact Sequences. These provide insight into the severity of future impacts and allow the Rx to recognize and mitigate the more severe problems first. The possible causes and their impacts are identified in a Suspect List, which is sent to the Rx.

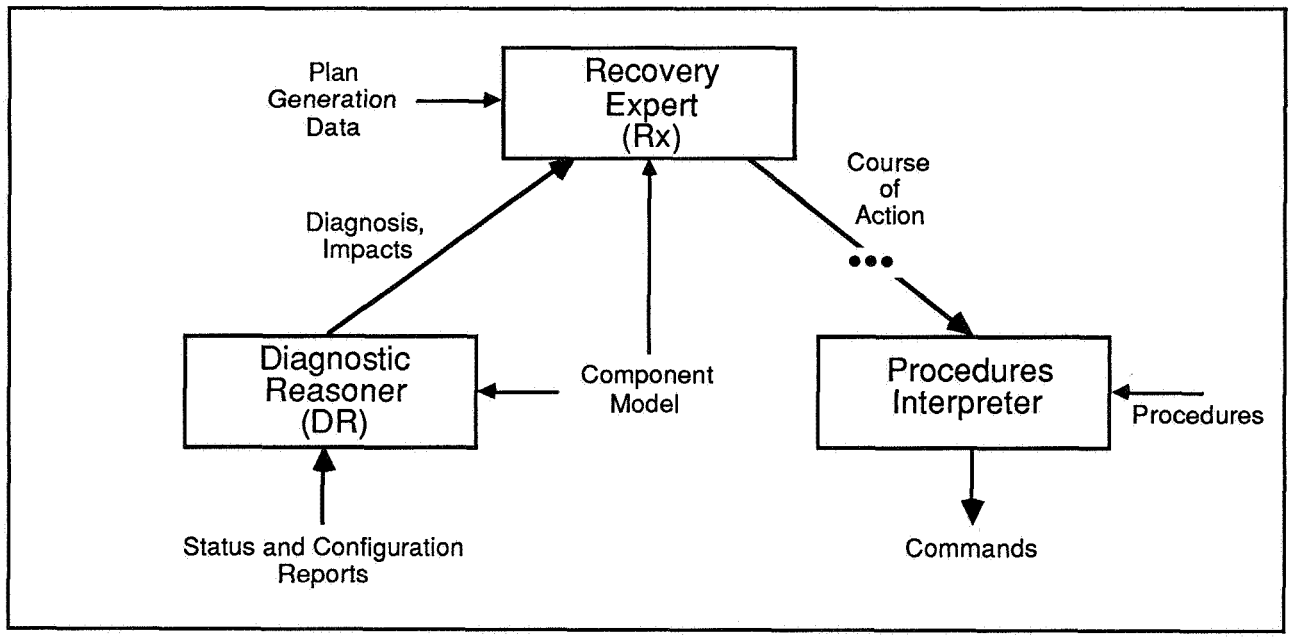

Figure 1. Overview of DR/Rx Process Flow

Rx Overview

The Rx bases its approach to the problem on supplied diagnoses and corresponding failure modes and effects criticality analyses. The Rx selects ways in which to attack the problem and develops partial plans that address specific aspects of the problem based on these chosen attacks. These partial plans are merged to generate comprehensive plans that address the entire problem. The Rx generates multiple plans, good and not so good, for further evaluation by the user. An overview of the processing performed by the Rx is presented in figure 2.

The Rx selects attacks on the problem based on the soundness of the diagnoses. When a diagnosis is sound the Rx attempts to develop primary plans that directly attack the root of the problem. When the diagnosis is equivocal the Rx attempts to generate primary plans whose intents are to collect information that could refine future diagnoses. For preliminary diagnoses the best response might be to do nothing. The Rx may also generate secondary plans that mitigate potential downstream impacts of the failure, allowing adequate time to realize the primary plans.

For each attack that is selected the Rx attempts to build partial Courses of Action. Initially, a top-level goal is generated based on the problem and the selected attack. This goal triggers the goal-directed planning process, which generates a tree-like structure of goals and activities. An activity specifies how to use a pre-defined procedure to achieve the goal of the activity. The procedures have prerequisites that lead to sub-goals, extending the tree. Partial Courses of Action are

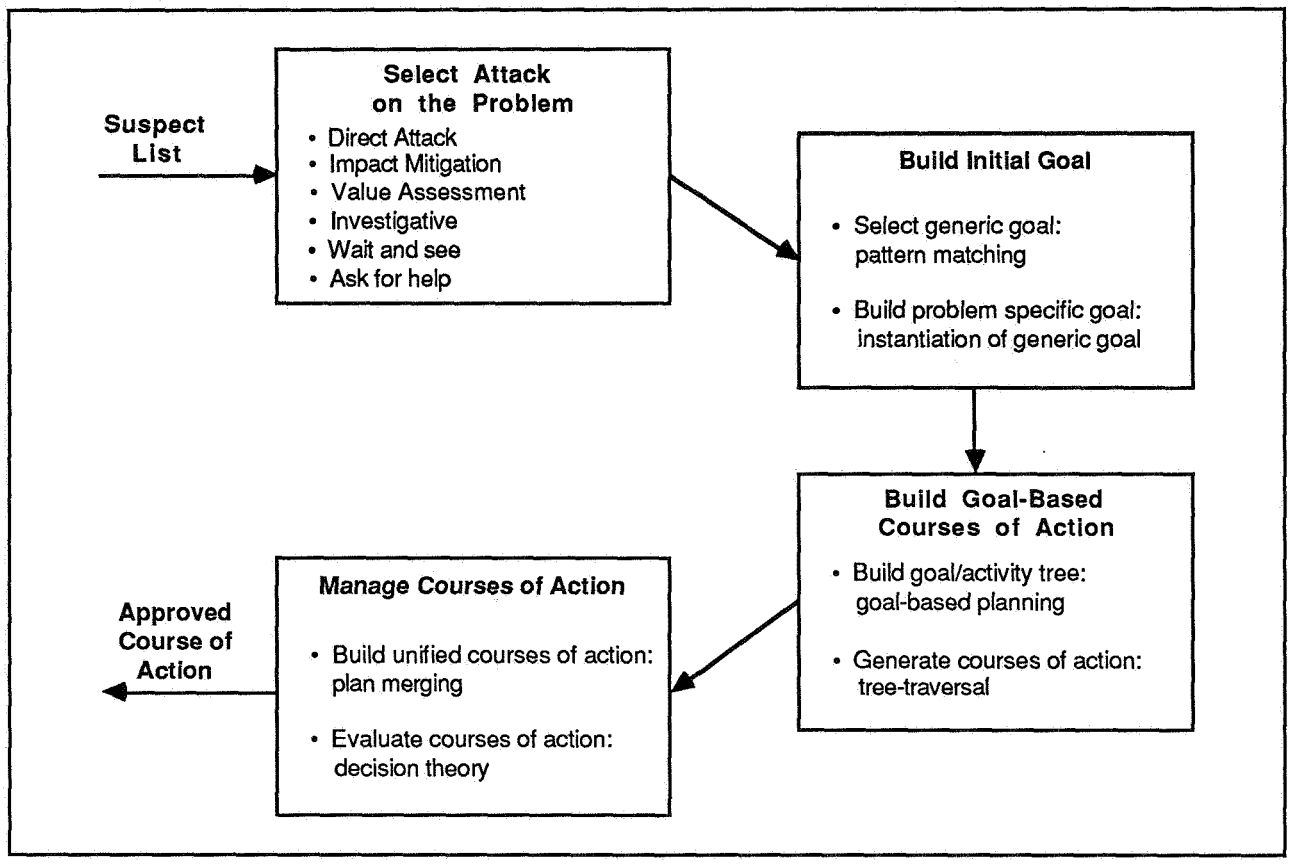

Figure 2. Overview of Rx Process Flow

extracted from this tree using tree traversal techniques. Follow-up actions that eliminate undesirable side effects of the activities can be added, again using goal-directed planning techniques.

Each partial Course of Action addresses only a part of the problem. The final action that must be taken before a recovery plan can be designated as complete is to merge partial Courses of Action, creating comprehensive Courses of Action that address the root of the problem, mitigate the side effects of the problem, and do so within the constraints levied by the problem.

Paper Overview

The above summary of what the Rx does provides little indication of how it does it. We intend in this paper to describe how the Rx interacts with external agents and how it produces Courses of Action in response to diagnosed faults. Since the DR and Rx are intended to reason about a dynamic system, we begin with a description of the mechanism used for representing and reasoning about relative time. In the following sections we describe the interactions of the Rx and external agents in terms of the data that are exchanged between the Rx and those agents, and we describe the means by which the Rx forms Courses of Action in terms of the processes used to build and evaluate the Courses of Action.

Relative Time Representation

The DR and Rx reason about a dynamic system where the state of the system changes with time. Timing and temporal relationships between detected events, projected future events, and planned activities are important to the functioning of the software. The problem here is not how to express the specific, absolute time at which detected events have occurred but how to express the relative timing between events: this is a matter of timing rather than time. We express relative time on a logarithmic scale of seconds. We chunk the time into discrete intervals by truncating to an integral value because this enables reasoning about time qualitatively and because this represents the limits of our accuracy. We terminate the scale at 0 since very small time intervals are not within the scope of the DR or Rx. This time scale is depicted in figure 3. An example of the use this scale is the time interval between an event and an expected impact of the event. If the time interval is 0 the impact is expected to be felt within ten seconds, and if it is 2 the impact should be felt in 100 to 1000 seconds. In discussions of our knowledge representation any reference to timing is made to this scale.

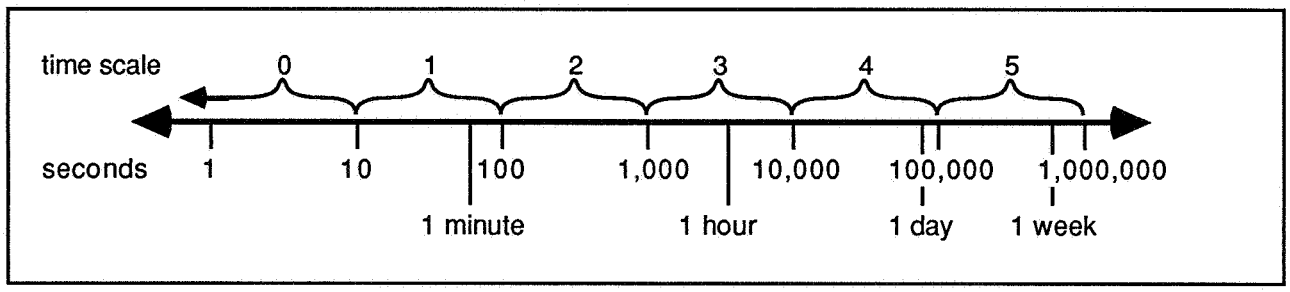

Figure 3. Relative Time Scale

Externally Visible Data

In this section we describe those types of data that the Rx exchanges with external agents: the Component Model, which describes the status and structure of *Freedom's* components; the Suspect Lists, which contain the diagnoses generated by the DR and cause the Rx to take action; the Procedure Metadata, which describe the actions that can be taken; and the Courses of Action that describe the actions to be taken.

The Component Model

The Component Model, which is at the heart of processing for the DR, also plays a significant role for the Rx. The Component Model incorporates dynamic structural and status information as well as other information that is not used by the Rx. The structural information describes the physical and functional relationships between components. Figure 4 illustrates two of the several types of inter-component relationships expressed in the Component Model. The status information describes the components' modes of operation, their health, and key operational values related to their behavior. Figure 5 illustrates some of the dynamic status information that is collected for a component instance. The Rx uses the structural and status information to tailor the Courses of Action it generates to *Freedom's* configuration and operating conditions. For example, the Rx would reject a Course of Action that involves the use of a specific component that had previously failed.

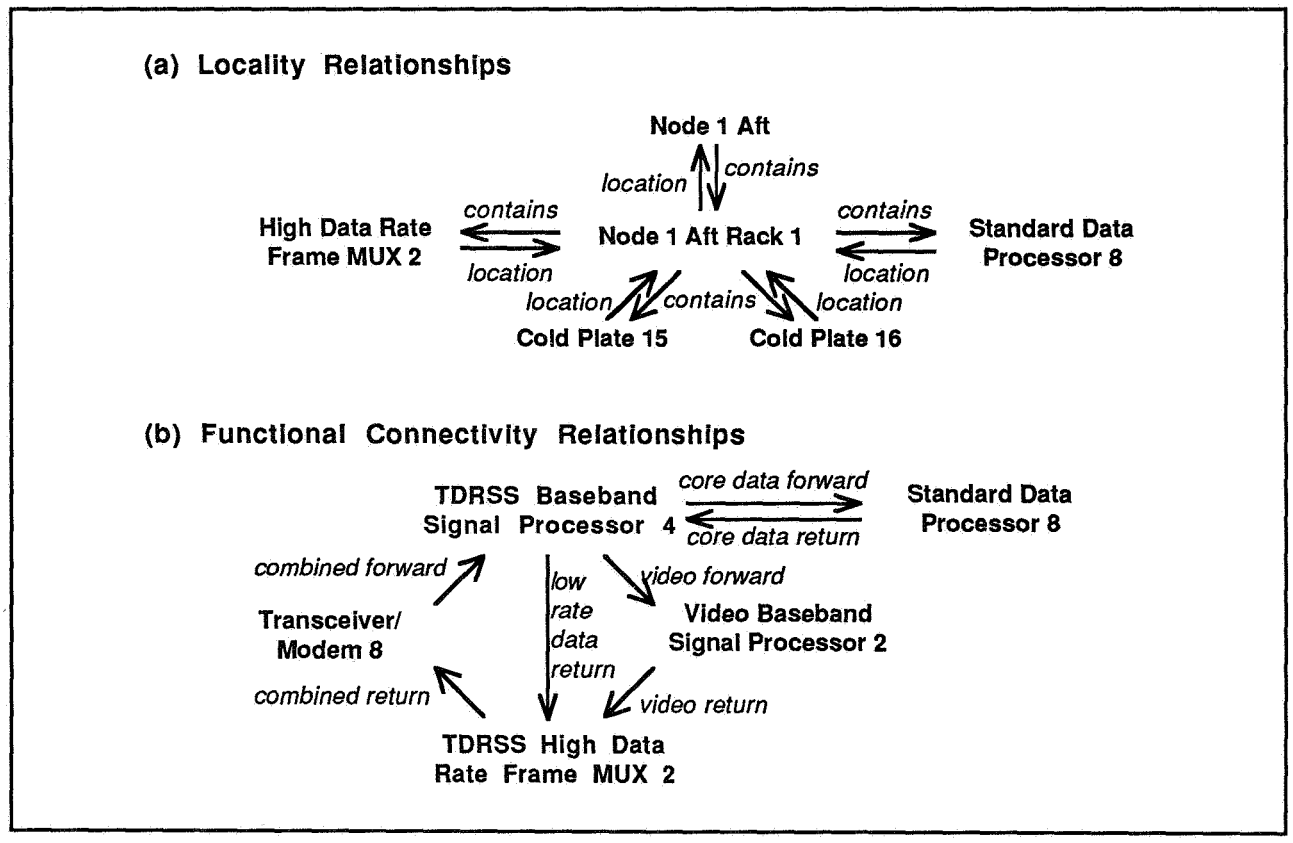

Figure 4. Component Model Structural Information

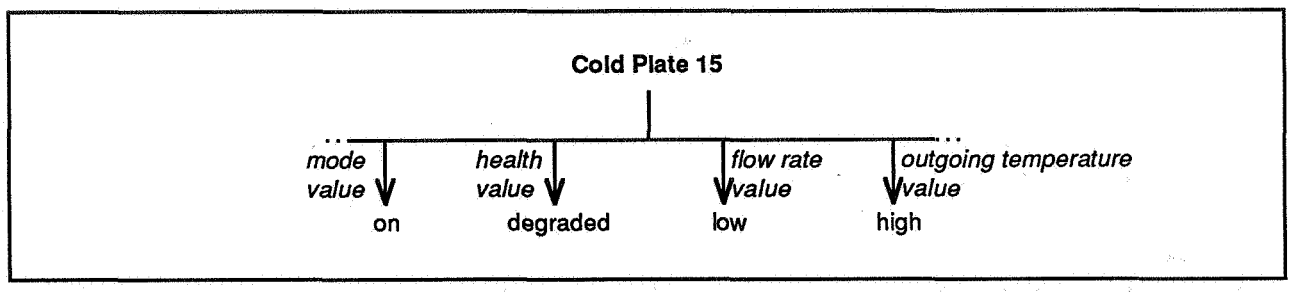

Figure 5. Component Model Status Information

The Component Model describes the components, the functions they provide, and where they are located. The components, functions, and locations are all described hierarchically, beginning with abstract object types that expand into more and more precise object types, and terminating in specific instances of object types. Each object class typically has many individual instantiations. This representation also encompasses information about characteristics of the components; we used inheritance techniques to reduce the amount of explicitly specified information.

Suspect List

A Suspect List identifies a set of related observed (off-nominal) behavior attribute values and those components whose off-nominal performance could result in those observations. Figure 6 shows the structure of a Suspect List.

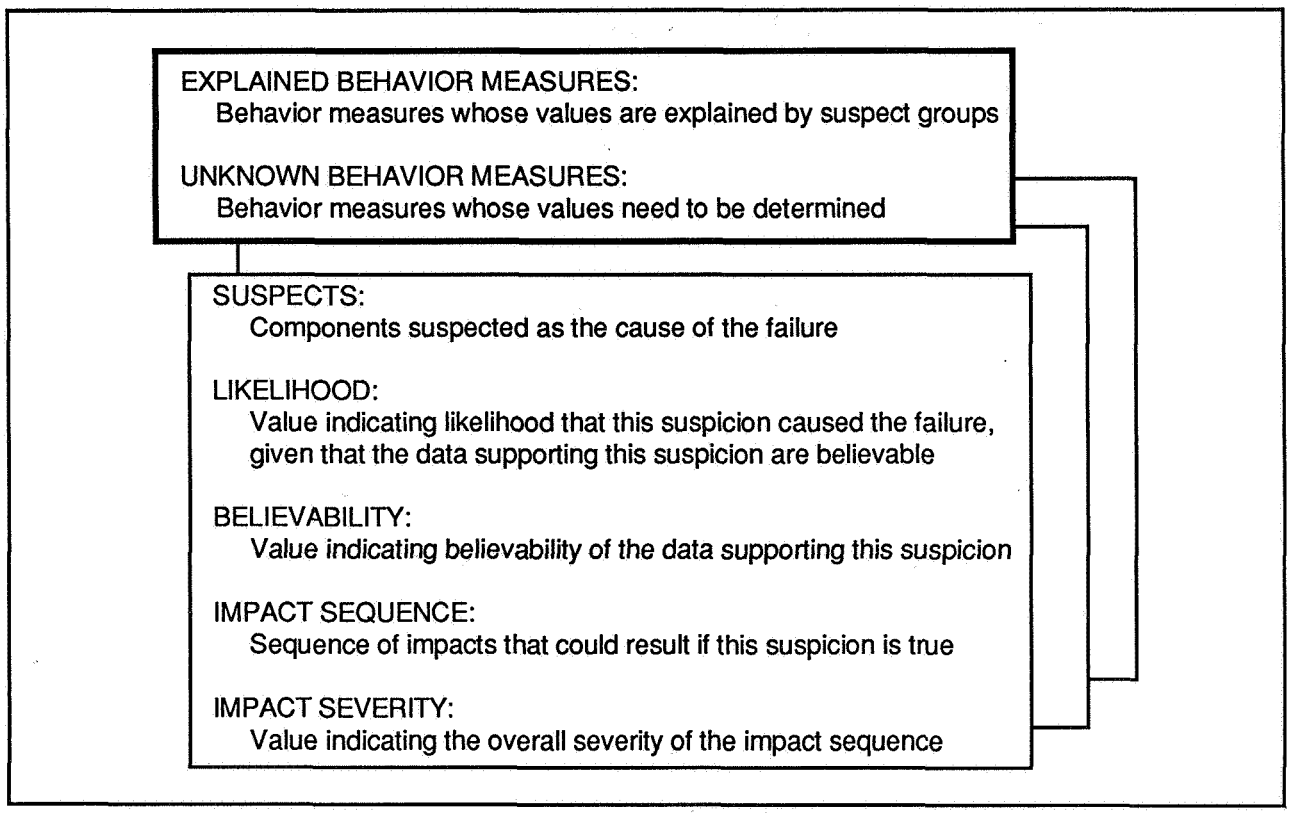

Figure 6. Suspect List Structure

A Suspect List is associated with a specific failure. The DR might find that more than one possible cause explains the same failure, in which case each explanation is reported as a separate diagnosis, or Suspect Group, in the same Suspect List. The DR might find that several independent component faults are required to explain the failure, in which case all the independent faults will be identified as a possible cause in a single Suspect Group.

A Suspect List also can identify key unknown behavior measures: operational behavior measures whose values are not available directly. The assessment of some behavior measures will require special resources (such as asking an astronaut to execute a procedure to collect data) or will induce inter-system interactions (such as taking a component offline to perform diagnostics). When the DR encounters one of these unknown measures along one of its diagnostic causal pathways it will post the measure in the Suspect List as an unknown measure whose assessment should help refine the diagnosis.

The DR adds likelihood and believability values and an Impact Sequence and its severity to each Suspect Group. Likelihood is an estimate of the probability that the Suspect Group caused the failure: for example, a component that fails frequently is more likely to be the cause than one that fails rarely. Believability indicates how reliable the data are that are used to determine the Suspect Group as the cause of failure: for example, some sensors are known to fail frequently or give unreliable readings. An Impact Sequence expresses the cascade of effects that might result from a given diagnosis. Each impact in the sequence is tagged with timing and severity information, and is an instance of a generic impact type. This allows the Rx to reason about the Impact Sequence generically and specifically, and allows the Rx to address the most acute or most severe impacts first. Figure 7 illustrates a portion of an Impact Sequence.

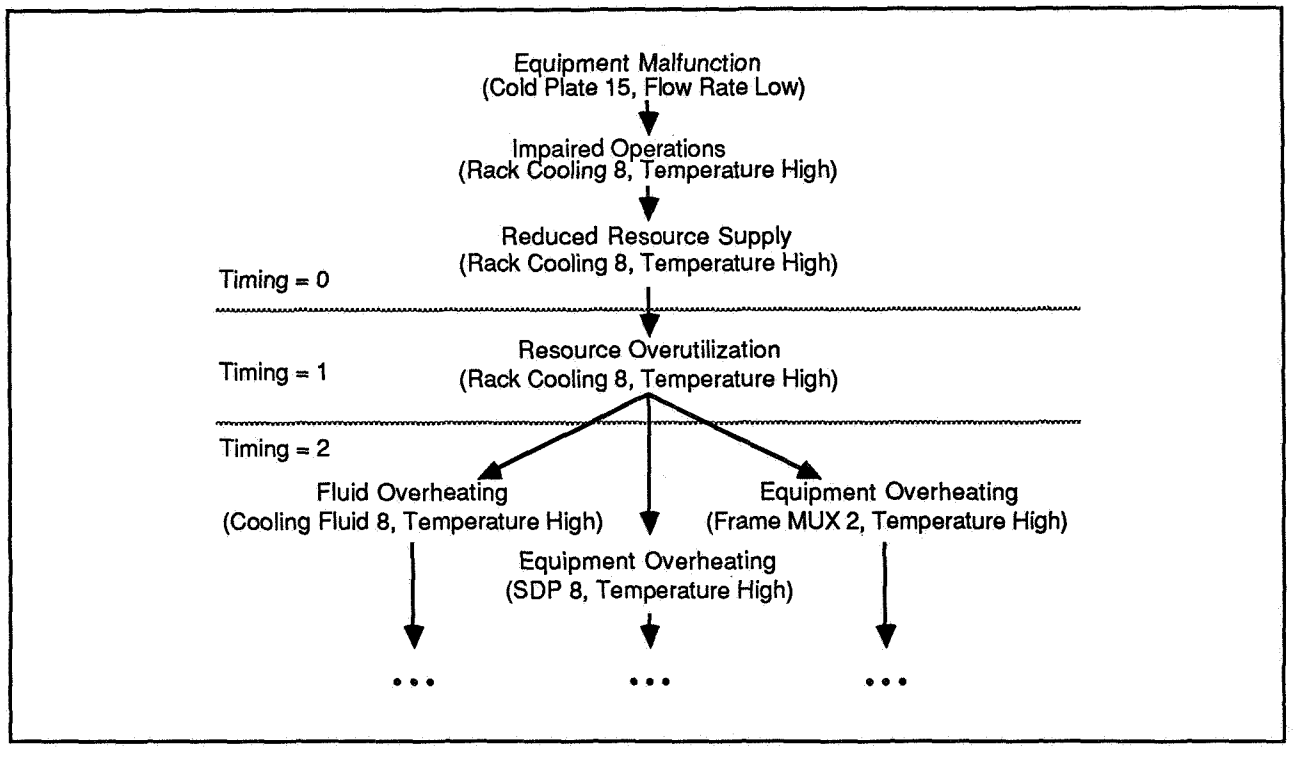

Figure 7. Impact Sequence Segment

Procedure Metadata

The Rx is able to build a Course of Action because extensive information about the pre-defined procedures is incorporated into its knowledge base. The Procedure Metadata describe how, why and under what constraints and conditions a procedure is used. These data are used to generate and evaluate a Course of Action. Examples of the Procedure Metadata are presented in table 1.

Table 1. Procedure Metadata

Procedure Name	Entry Conditions	Pre-requisites	Exit Conditions	Post-requisites	Motives	Schedul-ability	Timing
Test Cold Plate	Cold Plate Mode On	None	None	None	Cold Plate Health Suspect	3	2
Replace Cold Plate	Cold Plate Health not Nominal	Cold Plate Fluid level Empty	Cold Plate Health Nominal	Test Cold Plate	None	3	3
Drain Cold Plate	None	Cold Plate Mode Off	Cold Plate Fluid level Empty	None	None	3	2

Much of the Procedure Metadata is related to the goals associated with the procedure. A goal describes some aspect of the world state, such as the status of the health of a component. The entry conditions define what conditions must be true before a procedure is performed. The prerequisites define what conditions need to be made true before a procedure can be started. The exit conditions define what should be true when the procedure is complete (in other words, the goal of the procedure). The post-requisites define what needs to be done after completing the procedure. The motives of the procedures define what conditions would be true to lead one to believe that this procedure is one that applies. Schedulability is an a priori estimate of how long it will take before the procedure can start executing, while timing is an estimate of how long it will take to execute the procedure once it has started.

Course of Action

A Course of Action identifies the activities to be performed, temporal relationships between the activities, and entry conditions that should be true prior to performing the activities. These entry conditions provide a check against run-away operations: the execution of the Course of Action will be stopped if the entry conditions to an action are not met. Table 2 portrays a Course of Action for an equipment malfunction of a cold plate.

Table 2. A Sample Course Of Action

Procedure Name	Parameters	Temporal Relationships	Entry Conditions
Switch to Backup	Cold Plate 15	None	Cold Plate 16 Mode On
Valve Off Cold Plate	Cold Plate 15	Start after end of Switch to Backup	Cold Plate 16 Status In use
Drain Cold Plate	Cold Plate 15	Start after end of Valve Off Cold Plate	Cold Plate 15 Mode Off
Replace Cold Plate	Cold Plate 15	Start after end of Drain Cold Plate	Cold Plate 15 Fluid level Empty
Test Cold Plate	Cold Plate 15	Start after end of Valve On Cold Plate	Cold Plate 15 Mode On

Rx Processing

The Rx is responsible for building and recommending Courses of Action that, when executed, should result in recovery from the problem diagnosed by the DR. The Rx uses the pre-defined crew procedures as the elements from which it builds a Course of Action. A Course of Action includes actions that realize the recovery but also could include intermediate actions that mitigate the more acute consequences of the problem, providing adequate time to realize the recovery itself. A Course of Action should address the entire problem and should do this within the constraints levied by the problem. A future goal is to feed user-selected Courses of Action to the existing Procedures Interpreter prototype, which would drive simulations of various *Freedom* systems. This will provide a closed loop operation between the failure management and the systems prototypes, making the entire prototype operation more realistic.

The Rx performs four basic operations in its generation of a Course of Action, as depicted in figure 2: it selects attacks on the problem, it generates goals that address the problem in concert with the chosen attack, it builds Courses of Action that achieve these goals, and it manages and merges Courses of Action, resulting in user-approved comprehensive Courses of Action that address the entire problem. In the remainder of this section, we discuss the operations performed by the Rx to build the Courses of Action.

Selecting an Attack

Several options are available for dispositioning the diagnosis reported by the DR. The Rx must determine how to attack the problem based on the conditions in place at the time the failure occurs. The types of attacks that can be devised are direct attack, information gathering (value assessment and investigative), impact mitigation, wait and see, and ask for assistance.

Types of Attacks

The DR unequivocally and unerringly identifies some failures; in these cases, the problem can be addressed directly. For example, a known failed cold plate could be replaced by a crew member. In fact, a direct attack is the only approach that develops a Course of Action that solves the root cause of the problem. Even if other steps are taken to defer critical conditions from developing, eventually a direct attack on the problem must be taken to prevent manifestations of the problem from recurring again in the future.

Additional information could help the DR to refine diagnoses that are in doubt. The DR may indicate (in the Suspect List) that it needs to know the value of a behavior measure whose value is currently unknown; in these cases, the Rx will attempt to build a value assessment Course of Action whose intent is to determine the behavior measure's value. For example, a cold plate's flow rate may not be directly available but needs to be assessed, or a crew member may be asked to observe whether a specific component is exhibiting behavior that is observable by a person but not measurable by a sensor, such as whether a component is vibrating. Another approach must be taken when there are no unknown values that could obscure the cause of the problem; in these cases, there may be an existing diagnostic procedure that could help refine the diagnosis, and the Rx will use such a procedure to form the basis of an investigative Course of Action. For example,

a procedure that reconfigures the way that communications components are connected could be used as a basis for determining whether a given component is functioning properly.

The Rx is also concerned with assuring that the actions taken are sensible in light of the foreseeable impacts. When severe or acute downstream impacts could occur, the Rx will develop plans that mitigate these downstream impacts so that the desired action can be sensibly performed. For example, it may take a long time to replace a failed cold plate but moderate impacts such as damage to equipment cooled by the cold plate could occur in a short time and severe impacts such as the loss of a node sometime later. The Rx will develop plans (for example, equipment reconfiguration) to ensure that these impacts do not occur during the time needed to realize the recovery.

It might be imprudent for the Rx to begin generating plans when a preliminary Suspect List is reported by the DR: the costs of probably doing the wrong thing and of wasting valuable computer time to derive this unneeded Course of Action outweigh the benefits of possibly doing the right thing. In these cases, the Rx will do nothing, taking a wait and see attitude until the DR has observed some predicted near-term downstream impacts and revised its diagnoses.

Finally, the Rx will request operator intervention if it cannot find an adequate response. Since the DR and Rx are being implemented as decision support tools, the capability for the operators to interact with the processing is available at all decision-making points in the processing.

Approach

Selecting an attack involves evaluating the diagnosis according to a number of criteria and then weighing the alternatives based on these potentially conflicting evaluations. The criteria include

- the unequivocalness of the diagnosis;
- the immediacy and severity of the downstream impacts;
- the need for assessing an unknown behavior measure value;
- the cost of developing a Course of Action; and
- the surety of the diagnosis, based on
  - a statistical estimate of the likelihood of failure in the diagnosed failure mode(s) and
  - the believability of the reports that led to the diagnosis.

Techniques from decision theory (Markland, 1983; North, 1977) are designed to handle the quantitative nature of the evaluations that need to be made and some of these techniques explicitly handle making decisions under uncertainty. We use decision analysis techniques to determine the expected utility of an attack. Those attacks whose utility exceeds some preset value are released for Course of Action generation.

This thresholding allows an indeterminate number of attacks to be made. The intent of selecting an attack is not to select the single very best attack but to select the most appropriate attacks. Each of the possible attacks can yield partial Courses of Action that address a specific aspect of the problem. To form a comprehensive Course of Action that addresses the entire problem some of these partial Courses of Action would then have to be merged (see the section "Managing Courses of Action"). Thus there often is not a best attack; the best approach instead is to make several

separate attacks on the problem in a coordinated manner. Another aspect of managing Courses of Action is to select the best Course of Action; such a selection is best made when a reasonable number of options are available, and when the process of selecting attacks works well a reasonable number of options are generated for further evaluation.

Generating Goals

The Rx develops goals that address the failure in concert with the chosen attack. These initial goals trigger the goal-directed planner to build a partial Course of Action that achieves the goal. Several goals may have to be created and expressed as partial Courses of Action that are then merged to create a comprehensive Course of Action that addresses the entire problem.

Generic workarounds to impacts are used to generate goals for direct attacks and impact mitigation attacks. (The fault is always represented as the initial impact in the Impact Sequence, so a direct attack can be viewed as a special case of impact mitigation.) Selected impacts in the DR-generated Impact Sequence are matched with generic workarounds to generate goals concerning the exit conditions of procedures. For example, the Rx could address a reduction in cooling capacity to a rack as a special case of a resource supply reduction. A generic workaround for a resource supply reduction is to decrease resource utilization. Applying this generic goal to the specific problem results in the specific goal of reducing the cooling load at the affected rack. After the goals are generated, the Course of Action builder will look for procedures whose exit conditions meet the goals. Sample goal generation data are presented in table 3.

Table 3. Goal Generation Information for Impact Mitigation

Impact	Workarounds	Goal Generation Information
Equipment Malfunction	Repair or Replace Equipment	Equipment Health Nominal
Impaired Operations	Use Backup Capability	Backup Capability Status In-use
	Use Alternate Capability	Alternate Capability Status In-use
Reduced Resource Supply	Augment Resource Supply	Resource Level Nominal
Resource Overutilization	Augment Resource Supply	Resource Level Nominal
	Decrease Resource Utilization	Resource Consumption <= Resource Level

Each impact type can have several alternative workaround types. It is not necessarily the case, however, that each workaround type will be available for a given instance of an impact. For example, if there is a reduced resource supply of cooling, a workaround to that impact type is "Use Alternate Capability". In the general case of cooling, an alternate capability might be to turn on a fan when the air conditioner is not working to full capacity. In the case of a cold plate, however, there is no such alternate capability: a cold plate is the only source of cooling available. The Rx must determine the applicability of the workarounds to the situation being examined.

Goals are expressed symbolically (for example, Backup Equipment Status In-Use) in the goal generation data. Once a workaround is determined, the Rx creates a specific instance of a goal (such as Cold Plate 16 Status In-Use, where Cold Plate 16 is the backup unit to the cold plate exhibiting off-nominal behavior).

While a procedure's exit conditions are useful for finding a procedure with an applicable outcome, a procedure's motives are useful for finding a procedure that can be used when a given set of observations are true. The Rx builds goals for data collection and diagnostic attacks that cause the goal-directed planner to select procedures whose motives are to collect the required data items, to narrow down the number of suspect groups, or to verify an uncertain diagnosis.

Building Courses of Action

The Rx uses goal-directed planning (Chapman, 1987; Wilkins, 1989) to build a goal-activity directed graph. This graph does not explicitly specify the Courses of Action; these are extracted from the graph using graph traversal techniques.

Goal-directed planning matches goals with goal operators to build a plan. For the Rx, the goal operators are the pre-defined procedures. Unless a goal is already satisfied (the Rx checks for this in the Component Model) the Rx searches through the Procedure Metadata to find the procedure(s) that might accomplish the goal. Procedures can have prerequisites; when they do, this generates additional goals that need to be met, and the Rx once again searches the Procedure Metadata to find procedures that might accomplish these goals. A procedure can have more than one prerequisite, and these can be conjunctive, possibly requiring several procedures to accomplish them, or disjunctive, possibly creating alternate approaches to accomplishing them. The goal generation–goal accomplishment process iterates, resulting in the goal-activity graph, a tree-like structure of goals and activities, with the activities specifying how the operators are to be invoked. The graph building process obeys the constraints imposed by the failure and its impacts as well as constraints imposed by other operational conditions (such as configuration). Figure 8 shows a portion of a goal-activity graph built in response to an equipment failure of Cold Plate 15.

Even in our simplistic test application we found that an activity can simultaneously accomplish several goals. The basic goal-directed planning techniques outlined above would generate a separate activity for each goal. It would be foolish, and possibly erroneous, to perform the underlying procedure once for each goal. Instead, we devised techniques to identify such homologue activities—these are activities that have the same underlying procedure with the same parameters, which, when executed, will accomplish the goals of all of the homologue activities. Similarly, goals can have homologues, and we also devised techniques to identify such homologue goals; for example, multiple procedures might have the same prerequisite. Homologue activities or goals are not just identical but represent one single activity or goal (in comparison, identical twins are identical but are two separate individuals), so it is only necessary to process one of the homologues. A side effect of this is that the goal-activity graph is a graph rather than a tree.

The goal-activity graph does not directly specify a Course of Action. Graph traversal techniques must be used to extract the viable Courses of Action from the graph. The graph terminates in activities with no prerequisites, activities with unsatisfiable prerequisites, and goals that are already satisfied, and the Courses of Action for these are trivial. We extract the Courses of Action that achieve the top-level goal by building from the trivial Courses of Action at the bottom of the graph and working up to the top-level goal. The Courses of Action specify the activities to be performed and the temporal relationships between them. The procedures identified by the activities, when executed, should achieve the specific goal that the Course of Action addresses.

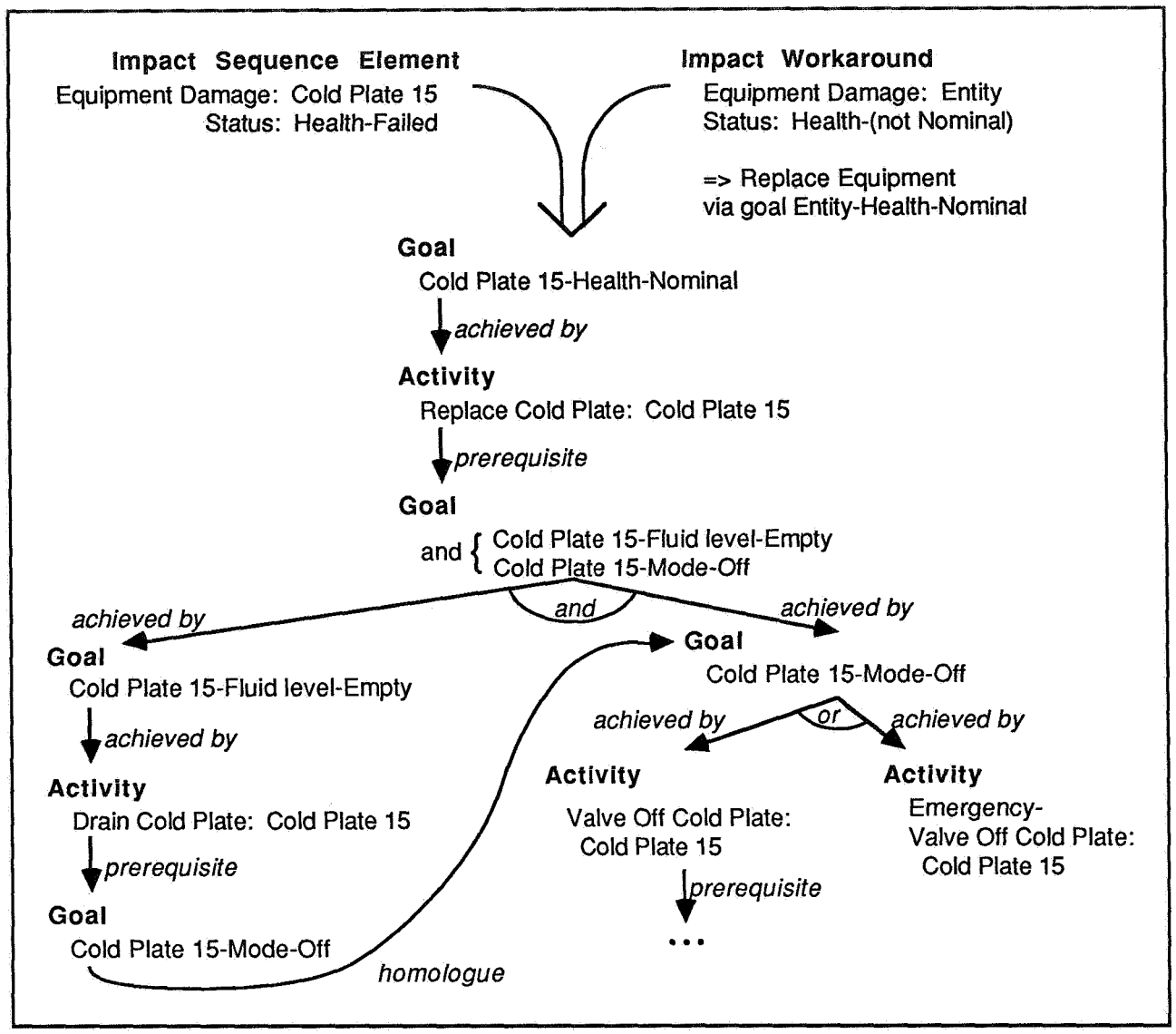

Figure 8. Portion of a Goal-Activity Graph

Developing a plan in a reasonable amount of time requires the judicious use of search and reasoning capabilities. The Rx uses procedure schedulability, timing, likelihood of success, and undesirability estimates to label some search paths as fruitless. The user can redirect the Rx to investigate these terminated paths.

Managing Courses of Action

The Rx can build multiple Courses of Action in response to a single problem. For example, the desired action should be achieved by a replacement Course of Action, but several impact mitigation Courses of Action will be required for this desired action to have a successful outcome. These multiple Courses of Action must be merged and ordered to form a comprehensive Course of Action

that addresses the entire problem rather than a portion of the problem. The attempt is to build Courses of Action that solve the root problem within the constraints levied by the failure.

The Rx attempts to develop several alternate means of addressing the problem. These alternate Courses of Action must be evaluated prior to execution. This final evaluation is performed using techniques from decision theory by a separately contracted prototype, the Technique for Rapid Impact Analysis and Goals Evaluation (TRIAGE) (Krupp and Burke, 1991). We have coordinated efforts with the TRIAGE implementors to allow for the integration of these functions.

Conclusions

The design of the Rx has been completed and an initial demonstration is available. Based on experience with this prototyping effort, including the Rx as well as earlier efforts from which it evolved, we have learned lessons in technology transition, software reuse, and prototype software life cycles. These lessons are discussed in (Baker, Hammen, Kelly, and Marsh, 1991).

This prototyping effort is not yet complete. The first step that we took in implementing an integrated prototype was to develop the DR and Rx as separate standalone applications. During this phase, we took care to express shared knowledge using the same structure. This phase has been completed, and initial demonstrations of the standalone DR and Rx are available. Our next step is to integrate the DR and Rx (over a network if necessary) to function cooperatively. Interfaces between the applications have already been defined; this step of integration merely implements these interfaces.

An important area of technical concern when implementing expert systems in a constantly changing environment has to do with coordinating past and current conclusions drawn by the software with the ongoing changes. Failures and their effects occur over a span of time. Consequently, the DR might identify suspects without complete knowledge of the problem, but it can update these diagnoses as additional information becomes available. The Rx might generate preliminary Courses of Action that can be overridden by Courses of Action generated in response to more accurate diagnoses. This association between current and past reports confronts most perpetually executing real-time monitoring systems, a class of programs to which the DR and Rx belong. The Rx does not yet take advantage of the correlations between the current and previously generated Suspect Lists; we plan to make the Rx utilize this information. We also plan to implement reuse of previously derived Courses of Action by extending the existing homologue detection capability.

The concepts underlying the DR and Rx have applicability beyond Space Station *Freedom,* the target environment for the initial prototypes. With this extensibility in mind, we separated the descriptions of the objects (such as procedures and components) from the generic capabilities of representing and reasoning about the objects. By formalizing this separation and by creating tools to develop and manage the application-specific objects we can create a failure management shell that could be applied to many domains that are hierarchical and are operated through pre-defined procedures.

Acknowledgements

The work referenced in this paper was jointly sponsored by the Automation and Robotics Section at NASA's Johnson Space Center under contract NAS9-18057 and by the MITRE Corporation through the MITRE Sponsored Research Program.

We thank Chris Marsh and Jayne Baker, our coworkers and the authors of the companion to this article describing the DR (Baker and Marsh, 1991), and Dennis Lawler, our NASA project monitor, who participated in the design of the Rx and critiqued its development.

References

Carolyn G. Baker, David G. Hammen, Christine M. Kelly, Christopher A. Marsh, 1991, *The Operations Management Application Failure Management Prototype*, WP-91W00006, The MITRE Corporation, Washington, D.C. (in press).

Carolyn G. Baker and Christopher A. Marsh, May 1991, "A Failure Diagnosis and Impact Assessment Prototype for Space Station *Freedom*," Paper presented at the 1991 Goddard Conference on Space Applications of Artificial Intelligence, Greenbelt MD.

David Chapman, July 1987, "Planning for Conjunctive Goals," *Artificial Intelligence*, Vol. 32 pp 333-377, reprinted in *Readings in Planning*, James Allen et al. eds., San Mateo, California: Morgan Kaufmann Publishers, Inc., pp 537-558.

David Hammen, Carolyn Baker, Christine Kelly, and Christopher Marsh, June 1990, "A Space Station Failure Management Prototype: DR and Rx," Paper presented at the 1990 Space Operations, Applications and Research Symposium, Albuquerque, NM.

Joseph C. Krupp and Thomas E. Burke, January 1991, *TRIAGE: Technique for Rapid Impact Analysis and Goals Evaluation*, DSA Report Number 84/1197, Decision-Science Applications, Inc., Arlington, VA.

Robert E. Markland, 1983, "Decision Analysis," *Topics in Management Science*, Second Edition, New York, New York: John Wiley & Sons, Inc., pp 790-824.

McDonnell Douglas Space Systems Company (MDSSC), September 1989, *Failure Tolerance and Redundancy Management Design Guide for Space Station Work Package 2 Systems and Elements*, MDSSC, Huntington Beach, CA.

D. Warner North, September 1968, "A Tutorial Introduction to Decision Theory," *IEEE Transactions on Systems Science and Cybernetics*, SSC-4:3, reprinted in *Readings in Uncertainty Reasoning*, San Mateo, CA: Morgan Kaufmann Publishers, Inc., pp 68-78.

David E.Wilkins, 1988, *Practical Planning: Extending the Classical AI Planning Paradigm*, San Mateo, CA: Morgan Kaufmann Publishers, Inc.