AEROSPACE REPORT NO.
TOR-2009(8591)-14

Effective Fault Management Guidelines

5 June 2009

Steven L. Hogan
Digital and Integrated Circuit Electronics Department
Electronics and Engineering Subdivision

Prepared for:

Space and Missile Systems Center
Air Force Space Command
483 N. Aviation Blvd.
El Segundo, CA 90245-2808

Contract No. FA8802-09-C-0001

Authorized by: Space Systems Group

Developed in conjunction with Government and Industry contributions as part of the U.S. Space Programs Mission Assurance Improvement workshop.

AEROSPACE REPORT NO.
TOR-2009(8591)-14

Effective Fault Management Guidelines

5 June 2009

Steven L. Hogan
Digital and Integrated Circuit Electronics Department
Electronics and Engineering Subdivision

Prepared for:

Space and Missile Systems Center
Air Force Space Command
483 N. Aviation Blvd.
El Segundo, CA 90245-2808

Contract No. FA8802-09-C-0001

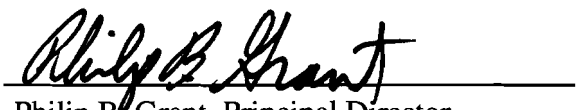
Authorized by: Space Systems Group

Developed in conjunction with Government and Industry contributions as part of the U.S. Space Programs Mission Assurance Improvement workshop.

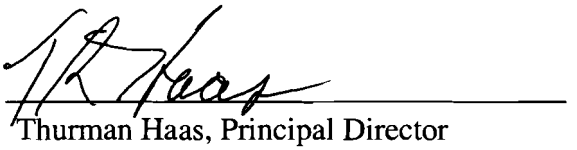
AEROSPACE REPORT NO.
TOR-2009(8591)-14

Effective Fault Management Guidelines

Approved by:



Philip B. Grant, Principal Director
Electronics Engineering Subdivision
Electronics and Sensors Division
Engineering and Technology Group



Thurman Haas, Principal Director
Office of Mission Assurance and
Program Execution
National Systems Group

Acknowledgments

This document was created by multiple authors throughout the government and the aerospace industry. For their content contributions, we thank the following contributing authors for making this collaborative effort possible:

Matthew Blanck—Integrity Applications Incorporated

Charles Channel—Raytheon

Mitch Homma—Integrity Applications Incorporated

Mark Kowaleski—NASA

Jeffrey Kurland—Boeing

Kendall Nii—General Dynamics

Alex Rubin—Lockheed Martin

Laurie Thiel—Integrity Applications Incorporated

A special thank you for co-leading this team and efforts to ensure completeness and quality of this document goes to:

Steve Hogan—The Aerospace Corporation

Carol Underwood—Northrop Grumman

Contents

Figures

Tables

1. Abstract

In an effort to develop a set of Mission Assurance Industry Workshop (MAIW) fault management methodologies and best practices, a review of the national space program history to identify fault management escapes/successes was performed. From this review, a collaborative government and industry team developed an initial set of fault management methodologies or "best practices" for implementing effective fault management throughout the program life cycle. This document provides the recommended fault management development tasks and system requirements, along with the guidance on how to perform these tasks, implement an effective fault management design, and perform an effective fault management verification program. Rationale and examples are provided when applicable.

This document is structured into two main sections: guidelines (section 4) and implementation suggestions (section 5). Sections 4 and 5 are organized in a roughly chronological order to correspond with the order that subjects would be encountered during the development of a fault management system during a program.

2. Scope

Space vehicle fault management consists of hardware, software, and procedural elements which collectively enable the space vehicle to continue operating, or reach a safe state, in the presence of faults. The extent of the resulting fault tolerance varies according to space vehicle requirements. The requirements should specify the type of autonomously-detected faults to be tolerated, as well the autonomous post-fault operational capability. Along with the reliability requirements, the fault tolerance requirements inform the amount of redundancy and cross-strapping provided by the space vehicle design. The required level of autonomy determines the extent to which this redundancy is employed by on-board responses as opposed to ground responses. It is important that these requirements and architectural decisions be made early in the space vehicle design process, so that the resulting fault management system can be comprehensive without being intrusive or excessively complex.

The contents of this technical operating report (TOR) are intended to be focused for use by an experienced technical person new to fault management to provide a broad background of items to be addressed in a space vehicle fault management design. The scope of this document addresses the on-orbit unmanned, near-earth orbiting vehicle and the single-fault aspects of fault management. Manned vehicles, deep-space vehicles, multiple-fault tolerance, and ground-based fault isolation and recovery are outside the scope of this TOR.

To implement an effective fault management system, it is important to establish a common understanding on what a fault management system is and what it is not. The role of a fault management system as part of an overall space vehicle mission assurance program is described below.

2.1 Role of Fault Management

What is fault management? There are many different definitions and there is no common terminology. To assist in developing a common terminology, Figure 1 shows the relationship of a few key definitions used in this TOR:

Failure—an unexpected response whereby the function is not recoverable.

Fault—An unexpected response whereby the function is recoverable, either by fixing it, managing around it, or by redundancy.

Fault Tolerance—The number of faults that the system must tolerate to meet its specifications. That is, a single fault tolerant space vehicle must still be able to meet its mission requirements after a single fault.

Fault Mitigation—The passive mitigation of faults through design. Examples are: ARM/EXECUTE command pairs for hazardous commands, extra solar array strings. Knowledge of the fault occurrence is not necessary to mitigate the effect of the fault.

Fault Protection—The implementation of design intended to actively control the effects of a fault.

For example, if the IRU fails, swap to redundant IRU. Fault protection requires knowledge that a fault has occurred (fault detection), and a response to contain the effects of the fault, so that the system will be able to still meet performance requirements. For example, if the IRU outputs zero, it must be failed, so swap to redundant IRU. The detection of the fault and subsequent response can be implemented autonomously on-board the space vehicle or by operators in the form of alarms and contingency procedures. Steps to return the space vehicle back into a configuration allowing continued operation to meet mission requirements

are included (fault isolation and fault recovery). Again, the implementation can be either autonomous on board or by ground operators. In safety terms, this is the step to control the effects of the hazard (AKA fault). This activity would also be known as resource management. It should be noted that the ability to control the effects of the fault is limited by the amount and type of resources employed.

Fault Management Systems Engineering—Fault Management An engineering discipline that address the occurrence of faults on space vehicle and provides a means for reducing their effect through cooperative design of both the space vehicle and ground elements and operator actions. Fault management includes fault tolerance, fault mitigation, and fault protection.

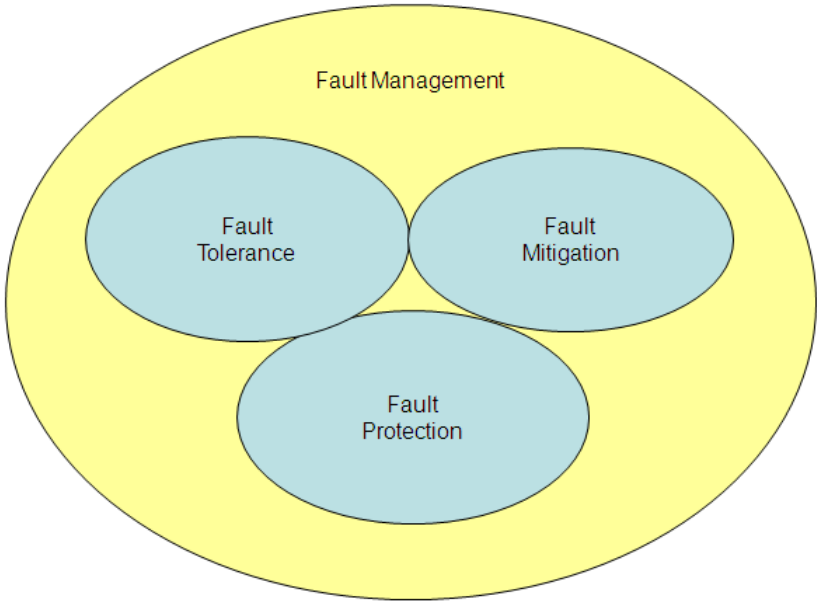


Figure 1. Key definitions relationship.

Fault protection is limited to the detecting, and responding to discrete defects which may arise in the operation of an otherwise healthy space vehicle. A typical fault would be the failure of a unit to perform to its specifications—e.g., a reaction wheel friction torque increasing due to bearing failure. Faults are thus a subset of anomalies, which include all deviations from expected/nominal performance. Anomalies (and faults) can include hardware faults, recoverable hardware upsets, infrequent extreme environmental conditions, or operator errors. Fault protection is limited to responses which involve switching out the failed component, and (if needed) reconfiguring the space vehicle to allow the failed component to be switched out— e.g., turning off the suspect reaction wheel. Fault management is intended to detect faults (or the effect of a fault), and intervene so that the fault does not permanently affect the safety of the space vehicle.

Fault protection complements the space vehicle's survivability design by responding to infrequent upsets caused by the space radiation environment. The space vehicle hardware should be designed to ensure that no irreversible damage results from operating in this environment. Because it is impractical to design the space vehicle hardware so that reversible upsets do not occur, these are properly addressed by fault protection. The maximum upset rates which fault protection is expected to handle must be negotiated between fault management and survivability/availability early in the space vehicle design.

Just as fault protection can be employed to address the infrequent upsets which are permitted by a sound survivability design, fault protection can also be employed to address other infrequent environmental occurrences. These include solar flares, which can blind CCD-based attitude sensors, or magnetic field anomalies which can overwhelm magnetic torquers. Since the environment is common to all redundant units, the space vehicle design must supply an alternative sensor or actuator for fault protection to switch to in this case, or operate temporarily in a degraded state.

Fault management can be employed to address operator errors. The space vehicle design should be such that no single inappropriate command permanently degrades the mission. Fault management can contribute to this objective. Beyond that, fault management can also provide the capability to respond to strings of inappropriate commands, or even to prevent these commands from being executed in the first place. These applications require that fault management function as an increasingly cognizant space vehicle manager, which brings with it an increase in complexity and operational intrusiveness. The authority assigned to fault management to intervene against ground commanding in this regard should be negotiated between fault management and flight operations early in the ground system design.

Fault management is not optimal for addressing defects in space vehicle unit design and cannot ensure protection against spacecraft design defects—e.g., a reaction wheel bearing failing due to being incorrectly sized for the launch environment. These defects include software coding errors. These sorts of design errors affect all like units, and so cannot be addressed by unit redundancy. The prevention of design errors must be addressed by a thorough analysis, design review, and validation program. If there is a concern about the comprehensiveness of this program, space vehicle unit design errors can be addressed through design diversity, in which multiple different designs for the same function are employed on a single space vehicle, - e.g. using an earth sensor and a magnetometer to perform degraded attitude determination until nominal attitude determination can be reestablished.

Fault management is not suitable for addressing defects in the space vehicle or unit architecture, specifically those which result in that inadequate separation of redundant functions. Fault protection relies on a redundant function being available in the event that the primary has been impacted by a fault. If the architecture is such that a credible single fault can affect both primary and redundant hardware, or a credible single fault can propagate from primary hardware to redundant hardware, then fault management will be defeated. Just as with the space vehicle units, these design flaws must be addressed by a thorough analysis and design review program.

Fault management is not suitable for addressing defects in the space vehicle assembly, either design or manufacturing related—e.g., a reaction wheel being installed with incorrect polarity. Although these sorts of integration errors can produce the same signature as a failure, they are more cost-effectively addressed by a solid ground test program. If the ground test program is constrained so that certain features of the design cannot be thoroughly validated before launch, then fault management should be designed to address the misbehavior of these features, although this can be a very expensive solution.

Fault management is not suitable for common cause defects. A common cause defect is one where a process, practice, or material results in a piece of the space vehicle (component, for example) that is deficient. In this case, the common cause defect defeats the intention of the redundancy (intended for random defects, not systemic defects). An example of a common cause defect is a part (component, sub-assembly, assembly, unit, etc) that has been overstressed during testing, where the overstress condition is part of a procedure and occurs to all parts processed, negating the premise on which the fault management is based.

Example: A pair of scientific satellites was launched in late 2000, and in less than three weeks both stopped receiving commands. Both spacecraft failed due to improperly implemented software, compounded by a fault-intolerant power-distribution architecture.

The Cause:

The root failure cause involved overheated relays bringing the receiver circuits down. The system requirements indicated that relays should receive pulsed commands. Software documents did not pick up this specification, and a constant voltage was supplied instead. A status indicator relay coil shorted under continuous heating in vacuum and caused the circuit breaker of Receiver B to trip. Receiver A should have been isolated from this fault, but was not because it was joined to Receiver B via an "OR" diode. It thus also suffered a current surge and blew the fuse, preventing the ground station from controlling the satellite.

The architectural oversight escaped design review probably because the status indicator relays were not thought to be crucial. However, because these relays drew current from both receivers, a short in either of them would cause a catastrophic failure of the system. The continuous command fault was not detected during unit test because the test set software correctly drove the relays with pulsed signals. System test should have caught the error because the continuously powered coils drew five extra watts, a considerable amount in a low-power system. Unfortunately, the extra power draw was not noticed [1].

2.2 Causes of Space Vehicle Failures

All space vehicles are subject to the hazards of their environment, handling practices, hardware and software failures as well as threats from adversaries (link denial). Understanding why and how satellites fail allows for selection of appropriate mitigation efforts, including architectural design choices and appropriate fault management techniques in order to maximize the chance of space vehicle survival. In this document a space vehicle hazard is defined as any real or potential condition that can cause damage to or loss of the space vehicle mission (primary or other functionality).

Faults may occur in any component of the space vehicle, or in the connections or interfaces between system components. These faults fall into two main categories:

1) Random hardware faults, assumed to occur with a constant random failure rate.

2) Common mode faults (including specification, design errors, improper parts and materials for the space environment, and workmanship errors).

It is important to recognize the existence of these two types of faults. A single random hardware fault will affect only one component or element of the system whereas a common mode fault may affect more than one component or element. This is of great significance when considering the use of redundancy [2].

Example:
In the late 1990s, at least four commercial satellites had problems with their spacecraft control processors (SCP), reportedly because whiskers grew on the relays and caused the power-supply fuses to blow. In three cases, both the primary and the redundant SCPs failed, and the satellites were lost.

Again in the late 1990s, three DOD programs incurred costly delays: one discovered tin whiskers in an atomic clock, the second found tin whiskers on ground lugs, and the third saw tin whiskers forming inside thin-film capacitors [3].

To better understand the causes of satellite failures, an analysis was performed by organizing documented failures from Aerospace TOR-2007(8617)-1 'Five Common Mistakes Reviewers Should Look Out For' [2] and a "War Stories" database of space vehicle anomalies [4]. Failures were divided into the following hazard categories, as shown in Table 1.

Table 1. Hazard Analysis Categories

Hazard Categories	Definitions
Hardware Faults	Random Part Failures
Systematic Faults (Common Cause)	All faults beyond hardware and software faults that can be introduced anywhere in the system life cycle including the design, implementation, operation, and maintenance phases.
Software Faults	Software Failures to include failures in requirements
Space Environment	ESD, Micrometeorites, space debris…
Engineering Fault (Parts & Materials)	Improper use of materials when designing for the environment
Engineering Faults (Design)	Systems Engineering Failures, Design inadequacies (not including materials), modeling errors, interface errors
Manufacturing Faults (Workmanship)	Failures in Parts Materials & Processes
Limited Engineering Knowledge	Failure of first attempt at new phenomenology
LV Failure	Failures resulting from LV failures up through separation to include low orbit insertion
Cause Unknown	Category for all other failures or anomalies

In this analysis, 325 space vehicle anomalies were reviewed using both sources mentioned prior and categorized producing the values in the Table 2.

Table 2. Space Vehicle Anomaly Totals

Hazard Categories	Aerospace LL TOR	War Stories	Totals	Percentages	
Hardware Faults	2	48	50	16%	100%
Cause Unknown	2	32	34	11%	
Systematic Faults (details below)	136	89	225	73%	
Software Faults	7	9	16	7%	100%
Natural Space Environment	2	13	15	6%	
Engineering Faults (Parts & Materials)	10	9	19	8%	
Engineering Faults (Design)	40	13	53	22%	
Manufacturing Faults (Workmanship)	46	34	80	33%	
Limited Engineering Knowledge	1	9	10	4%	
Launch Vehicle Failure	37	11	48	20%	

Figure 2 shows the results of this analysis. Figure 3 shows the breakdown of the systematic (common cause) faults.

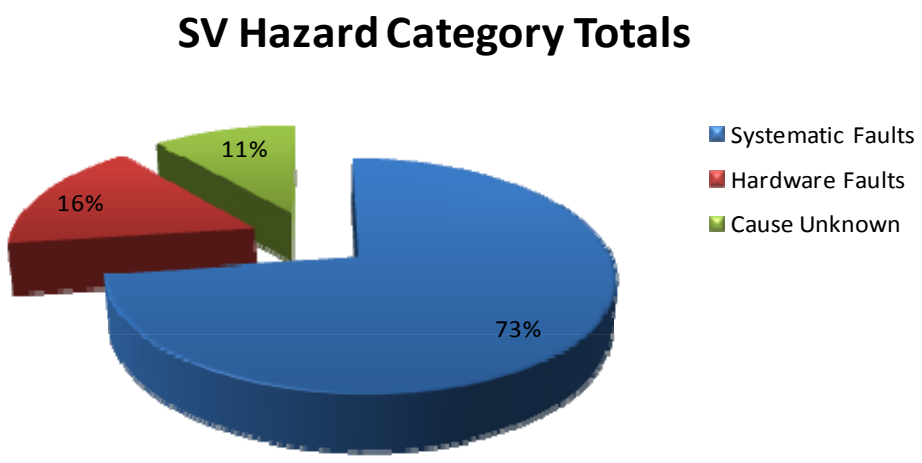


Figure 2. Space Vehicle hazard category totals.

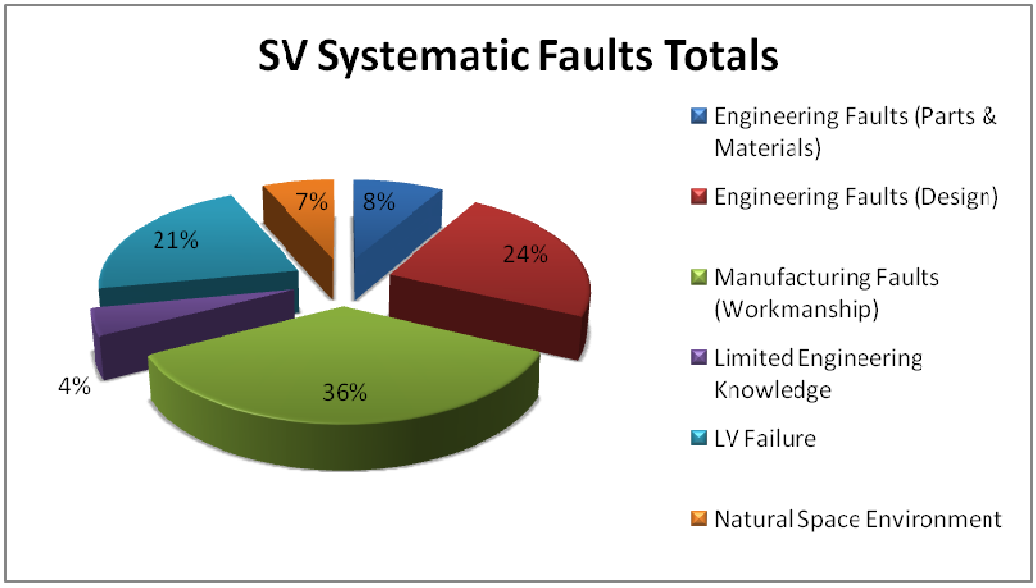


Figure 3. Space vehicle systematic faults.

The results of a 1992 survey categorizing approximately 2,500 spacecraft failures between 1962 and 1988 is provided in Table 3 [3]. Table 4 shows the distribution of the failures by subsystem.

Table 3. Survey of Failures from 1962 —1988 by Category

Types of Failure

Assigned	%
Design	24.8
Environment	21.4
Operations	4.7
Random	**%**
Parts	16.3
Quality	7.7
Other	6.3
Unknown	18.9

Table 4. Survey of Failures from 1962 — 1988 by Subsystem

Distribution of Failures

Space vehicle Bus (total 73.3)	%
Telemetry, Command and Control	24.6
Guidance and Navigation	13.6
Electric Power	13.2
Data System	9.1
Thermal Control	5.6
Propulsion	3.7
Structures	3.5
Payload (total 26.7)	**%**
Visual/IR Sensors	13.1
Communications Payload	5.2
Special Payloads	4.9
Navigation Payloads	3.5

To better compare the above survey data with the space vehicle hazard category data, the data in Table 3 was put into pie-chart format and is shown in Figure 4.

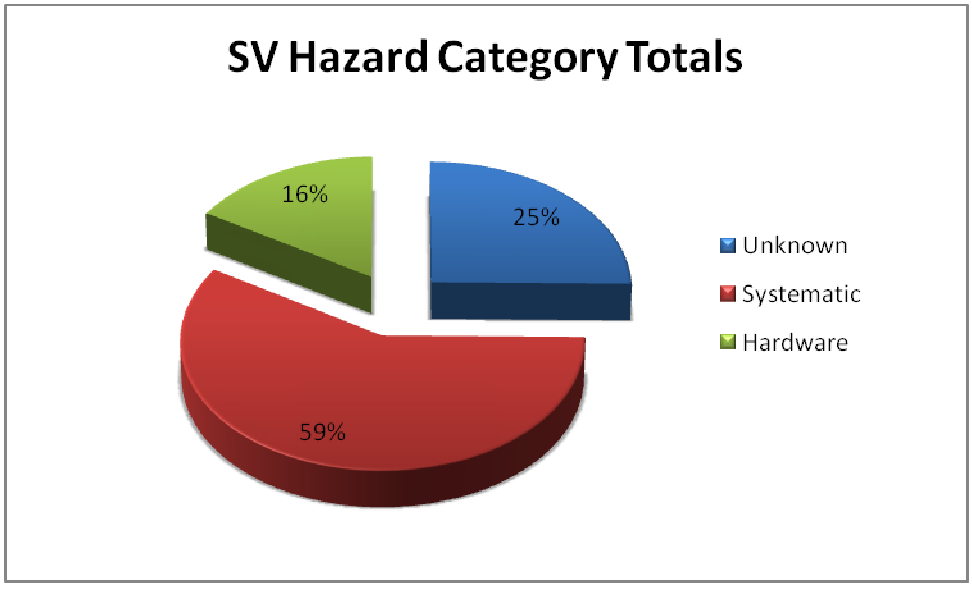


Figure 4. Space vehicle failures by category (prior to 1972).

The categorization of the failures is somewhat subjective, and depends on context. The data presented attempts to categorize failures based upon the interpretation of root cause. In both analyses, environmental faults (i.e. anomalies from radiation) were considered design errors, in that the space vehicle should be designed to survive its expected environment.

The space vehicle hazard analysis shows that most space vehicle failures are due to common cause, or systematic faults rather than random hardware faults. To maximize the chance of space vehicle survival, fault management systems should encompass both robust space vehicle design, as well as a fault management system.

There should be a focused effort on preventing common cause failure mechanisms. Analysis of the space vehicle failure data reveals that 60% of the failures are due to systematic faults relating to design errors (and it could be argued that the space environment failures could be considered design faults), component, and workmanship issues that fault management systems are not intended to address. To ensure mission success, it is imperative that the space vehicle be designed and built to withstand the effects of the space environment.

2.3 Fault Management Systems

A *fault management* system addresses *faults*. The reaction to a fault may be active or passive. An *active* reaction includes some manner of space vehicle reconfiguration to accommodate the fault, typically by exchanging offline and online resources. A *passive* reaction takes advantage of pre-existing online redundancy.

Whether active or passive, the reaction to a fault typically includes two phases. The focus of the first phase is the containment of the fault. This phase is termed fault detection and response and is responsible for the immediate reaction to the fault, ensuring that the fault does not propagate and that the space vehicle can continue to operate in a safe manner. The second phase is termed fault isolation and recovery and is responsible for optimizing the space vehicle (and ground) configuration to support the remainder of the mission.

Fault protection consists of *fault detection monitors* and *fault responses*, which detect faults, and then reconfigures the space vehicle to contain the fault. Fault protection may be implemented on-board or in ground procedures, depending on space vehicle autonomy requirements. On-board fault protection is required for faults that require rapid response, or for faults which interfere with the ground's ability to communicate with the space vehicle. Other faults may be addressed by ground-based tests (alarms) and responses (contingency procedures).

3. Bibliography

The following applicable documents contain additional guidance material.

Christopher, D. A., & Hoheb, A. C. (2004). Systems Engineering Charter for National Security Space System Programs. The Aerospace Corporation.

NASA Fault Tree Handbook with Aerospace Applications (2002),

The Aerospace Corporation (2006). Space Vehicle Systems Engineering Handbook TOR-2006(8506)-4494

4. General Fault Management Guidelines

The guidelines provided herein define the fundamental requirements for sound and effective fault management systems. There are eight main, recommended fault management system-development tasks. The following paragraphs provide the recommended requirements for each of these tasks. Where applicable, the rationale for the recommended requirement is provided. Examples are also included when available, illustrating the benefit of the recommended requirement.

4.1 Fault Management System Development

All space vehicle programs should integrate independent fault management into the systems engineering process. The overall fault management development approach should be defined during the program pre-acquisition phase by the acquisition team's fault engineer point of contact (POC). This development approach should include an independent fault management organization or system engineering POC on both the contractor's and customer's staff with defined roles and responsibilities, a forum for coordinating the fault management tasks, and the fault management program milestones. Ideally, fault management responsibility should be part of the program's systems engineering organization to better allow coordination with the other systems engineering disciplines, space vehicle subsystems and the ground segment.

4.1.1 Fault Management Development Tasks

The fault management systems engineer(s) should be responsible for the following tasks:

4.1.1.1 Fault management CONOPs Development

The fault management systems engineer should participate in the development of the space vehicle concept of operations (CONOPs) and be responsible for defining the fault management CONOPs and participate in the associated trade studies.

4.1.1.2 Fault Management Working Group

The program should establish a working group for the fault management systems engineer to coordinate the execution of the fault management development tasks described herein.

Rationale: A forum is needed to coordinate with program management, radiation survivability, reliability, subsystem engineers, integration and test, ground segment and the customer during the development and verification of the fault management system.

The fault management systems engineer should be responsible for coordinating with program management, radiation survivability, reliability, subsystem engineers, integration and test, ground segment and the customer during the development of the fault management system. The fault management working group provides a forum for this coordination. The fault management working group or equivalent should continue on an as needed basis throughout the program life cycle to provide a forum for addressing faults identified after the major design activity has completed.

The fault management systems engineer should coordinate with the various other system engineering (Mission Assurance) disciplines during the space vehicle architecture design phase and for fault identification. Figure 5 shows the relationships between the fault management systems engineer and other systems engineering disciplines.

Coordination is particularly needed with radiation survivability. The fault management systems engineer needs to know what failures can occur as a result of radiation exposure and to should work with radiation survivability to ensure any residual failures due to radiation can be effectively dealt with using fault management or if further measures are needed to eliminate the effects of these failures.

The fault management systems engineer also needs to coordinate with reliability engineers. The amount of hardware redundancy is usually driven by program reliability requirements. As discussed in section 5.4.2 (Hardware Redundancy), the amount and type of redundancy used affects the fault management system design. Reliability also has a role in fault identification, in that many times, it is the reliability organization that performs the FMEA.

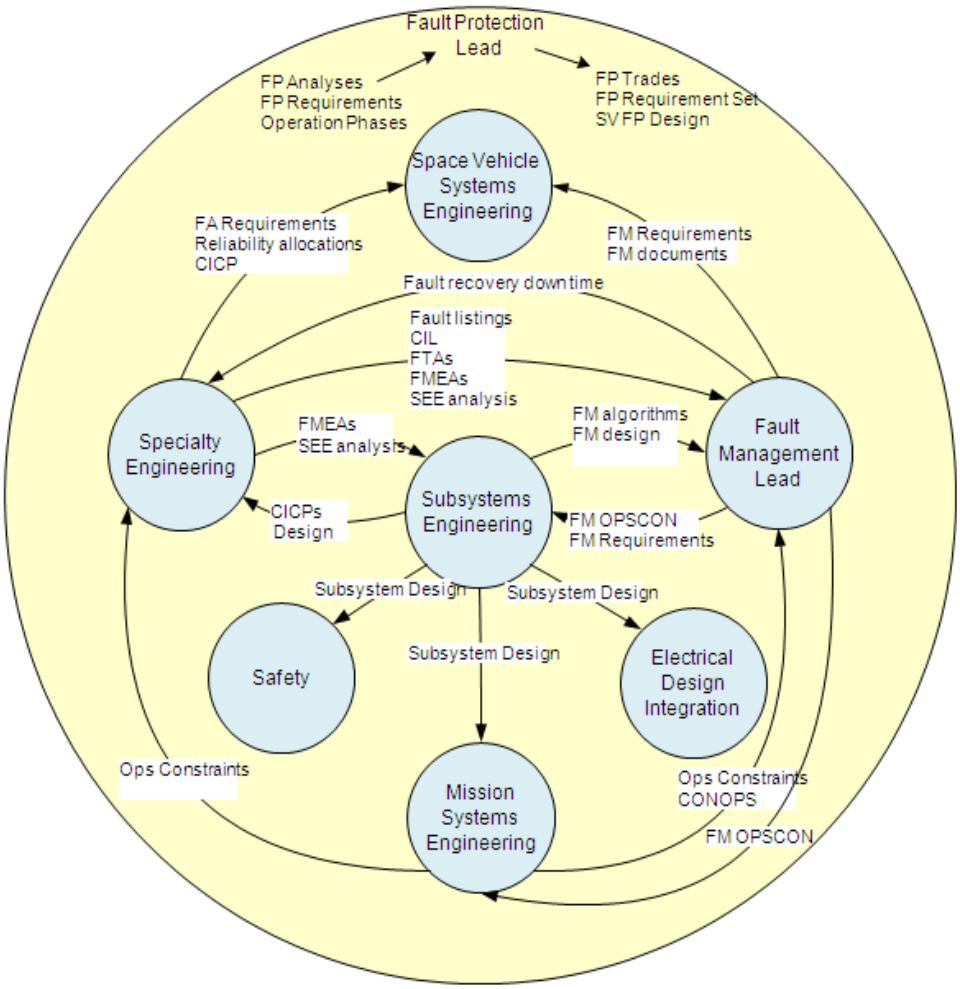


Figure 5. Fault management relationship to system engineering disciplines.

4.1.1.3 Fault Management System Requirements Definition

The fault management systems engineer should be responsible for the definition of the system specification (unless provided), fault management system requirements, the development and flowdown of these requirements to the space and ground segments, space vehicle subsystems and units. These requirements include fault management modes, system disposal and fault tolerance requirements.

An effective fault management system is not limited to just bus subsystems. Payload faults and subsequent fault management should also be considered. The payload is just as much a part of the space vehicle as the bus subsystems.

Space vehicle reliability, survivability, availability, fault tolerance, fault mitigation, and fault management requirements should be grouped together in the mission specification. The fault management requirements should address all modes (deployment, normal, safe-hold, etc)

4.1.1.4 Space Vehicle Architecture Definition

The fault management systems engineer should participate in the development of the space vehicle architecture. The overall space vehicle architecture has a significant effect on the fault management architecture and complexity of the fault management. The fault management systems engineer should be responsible for ensuring that the space vehicle architecture supports all of the fault management requirements.

4.1.1.5 Fault Identification

The fault management systems engineer should be responsible for defining the fault identification methods to be used, for performing any system level fault identification analysis, and for documenting the fault identification results.

The fault management systems engineer should be responsible for defining the method to be used to show fault coverage, that is, show that identified faults are either mitigated or protected, and for documenting fault coverage analysis results.

The fault management systems engineer should coordinate with the payload, subsystems, and unit engineers to identify any hazardous (to the hardware) constraints, potential failure mode and faults that could potentially result in the loss of mission. See section 5.2 (Fault Identification Methods) for descriptions of effective fault identification methods.

If a probabilistic risk assessment is required, the fault management systems engineer should coordinate with reliability to perform this task. See section 5.3 (Risk Assessment Methods) for details.

4.1.1.6 Fault Management Requirements Development

The fault management systems engineer should coordinate with the space vehicle subsystems, including payloads, to define the fault management modes and the specific actions needed for each mode to meet the recommended requirements contained in section 4.1.3.3 (Fault Management Mode Guidelines). Fault management mode implementation guidelines are provided in section 5.4.4.4 (Space Vehicle Fault Management Modes).

The fault management systems engineer should coordinate with the appropriate space vehicle subsystems, including payloads, to allocate the recommended fault management requirements contained in section 4.1.3.4.1 (Fault Mitigation Requirements Allocation) to the appropriate subsystems and units. It is important for the fault management systems engineer to know what fault mitigations are in place before designing the autonomous fault management monitors and responses. If the effects of faults can be eliminated or reduced by the fault mitigation, then there will be fewer faults that fault management needs to contain, simplifying the fault management design.

The fault management systems engineer should participate and coordinate with the subsystems engineers to define and develop specific fault monitors and fault responses to correct the effects of the identified faults. These detailed requirements include both hardware and software implemented fault protection. Recommended fault protection requirements are contained in section 4.1.3.4.9 (Autonomous Fault Protection Guidelines). The resulting fault management system design, fault detection monitors, fault responses, and space vehicle modes depends upon the mission, space vehicle architecture, fault management architecture, the amount of needed autonomy, and the identified faults. Section 5.4.4.1 (Autonomous Fault Protection Architectures) describes the various architecture choices and how these architectures affect the fault management system.

The fault management systems engineer should coordinate with the appropriate space vehicle subsystems to allocate and derive the necessary lower-level requirements needed to meet the recommended diagnostic data requirements contained in section 4.1.3.4.10 (Diagnostic Data Guidelines). In addition to solid-state data recorders, diagnostic data can be stored in on-board processor memory. Anomaly data storage should be stored at a high a rate as possible. Normal state of health telemetry can be stored at a low telemetry rate. Circular telemetry buffers can also be used to store anomaly data at a high rate, so long as the data immediately following the anomaly is not overwritten.

The fault management systems engineer should coordinate with the appropriate space vehicle subsystems to allocate and derive the necessary lower level requirements needed to meet the recommended system disposal requirements contained in section 4.1.3.3.8 (System Disposal Guidelines). Many times safe mode can be used to implement system disposal.

The requirements flowdown activity should include any analysis needed to derive lower-level requirements from the system specification. Fault management system requirements need to be allocated and traced down to the lowest-implementation level in hardware and software.

4.1.1.7 Fault Management Design Reviews

The fault management systems engineer should hold specific fault management design reviews to ensure sound fault management requirements, fault management design, and fault management verification.

Fault management design reviews can be formal or informal so long as careful review of the fault management design (architecture, fault monitors and fault responses) is performed to ensure that the design meets the requirements and will perform as expected when needed [5]. Fault management design reviews should be incorporated into the program milestones.

4.1.1.8 Fault Management Verification and Selloff

The fault management systems engineer should be responsible for fault management verification and validation activities. This effort includes definition of the fault management verification plan, ensuring completeness of all lower-level verification planning and products, and documenting the verification evidence. The fault management verification evidence should include fault coverage analyses documenting the methods used to either mitigate or protect the identified faults, and the verification data showing that the fault management system will perform as expected if needed during mission operations, including activation and critical events.

The fault management systems engineer should coordinate with payload, subsystems, and unit engineers to define and develop the complete fault management verification plan. Details on fault management verification methods are described in section 5.5 (Fault Management Verification Techniques).

Fault management sell-off can be through a sell-off review, formal requirements sell-off, or both. A sell-off review allows for cross-discipline and program management review of the final fault management verification data. Formal sell-off of the fault management requirements allows for a thorough review of the fault management verification data to help ensure that fault management system will perform as expected if needed.

4.1.2 Fault Management Program Milestones

The program should establish the fault management milestones and relate these to the major program milestones, program element responsibility, and required entry and exit criteria.

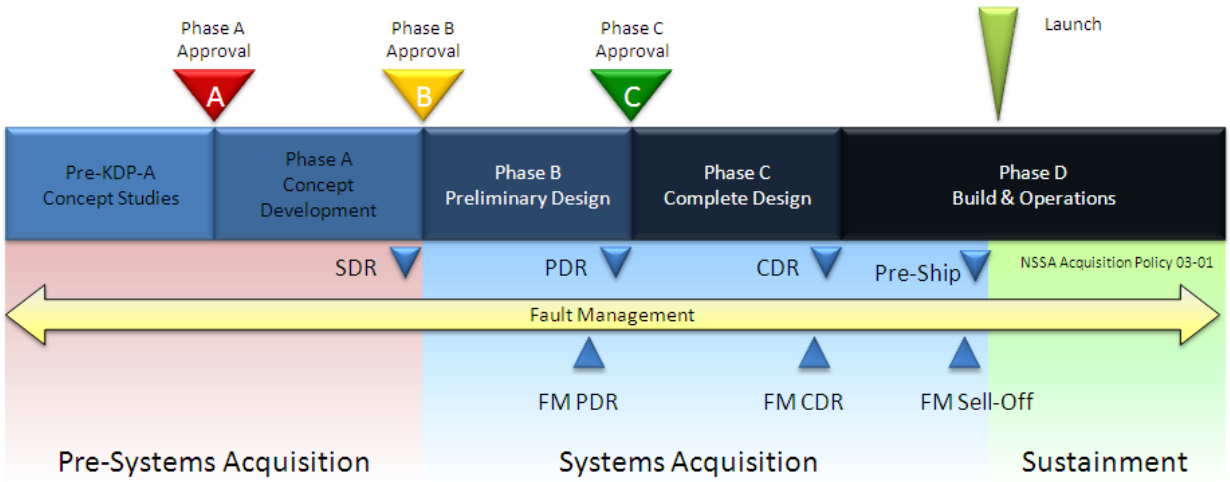


Figure 6. Fault management program milestones.

The fault management systems engineer should be responsible for participating in the major program reviews (SRR/SDR, PDR, and CDR) to ensure that the program fault management requirements will be met [5]. In addition to the program reviews, formal fault management design reviews (PDR, CDR) are recommended to ensure that the fault management system design undergoes the same level of scrutiny as other subsystems.

4.1.2.1 Pre- Acquisition

The following items should be defined prior to program acquisition.

- Fault management systems organizational relationships
- Fault management system coordination forum
- Fault management system program milestones and reviews identified
- Fault management system requirements defined consistent with any applicable space vehicle risk class definitions.
- Fault identification method defined

4.1.2.2 Design Reviews

This section defines the minimum entrance criteria for the various milestone reviews that should be conducted. Additional fault management reviews may be conducted.

4.1.2.2.1 Program Design Reviews

4.1.2.2.1.1 System Requirements Review (SRR)/ System Design Review (SDR)

The following items should be completed and demonstrated at the program SRR/SDR.

- Fault management CONOPs defined—Includes needed diagnostic telemetry, speed of ground reaction to faults and anomalies for determination of how much autonomy is required, amount of ground interaction allowed in fault correction actions.
- Review of previous programs lessons learned
- Fault management requirements allocated to the space and ground segments
- Fault management modes defined
- Autonomous fault protection architecture defined
- Fault identification methods defined
- Fault management verification plan defined
- Schedule

4.1.2.2.1.2 Preliminary Design Review (PDR)

By PDR, the fault management system preliminary design should show that all required failure conditions have protection or detection/correction requirements defined. The requirements flow-down should be presented that define the appropriate hardware and software diagnostic telemetry, stored data requirements, and the ground responsibilities/reactions. The fault management system architecture should be defined by PDR [5].

4.1.2.2.1.3 Critical Design Review (CDR)

By CDR, the fault managements detailed design should show responses to failure modes and effects analysis (FMEA). The interference analysis for fault thresholds and persistence should be complete. The fault management design should be supported at CDR with scenario simulations plus HW and SW testing [5].

4.1.2.2.2 Fault Management Design Reviews

Multiple reviews of the fault management are needed to ensure that the fault management design, implementation, and verification are appropriate and remain consistent with the space vehicle implementation. The best fault management design in the world is no good if it does not reflect the *implementation* of the space vehicle. As the space vehicle design matures, changes are made, the fault

management design, implementation, and verification must change to reflect the as-built system fault management is intended to protect. The recommended reviews and content is provided in Figure 7 and discussed in the paragraphs below.

The Fault Management systems engineer should set the fault management program review milestones dates to correspond to the work flow, staffing plan, and budget. In the reality of a real budget and schedule, the Fault Management systems engineer need to plan what content will be used to satisfy each program milestone and set the commensurate expectations for each milestone. Ownership of the milestones and what they represent is key.

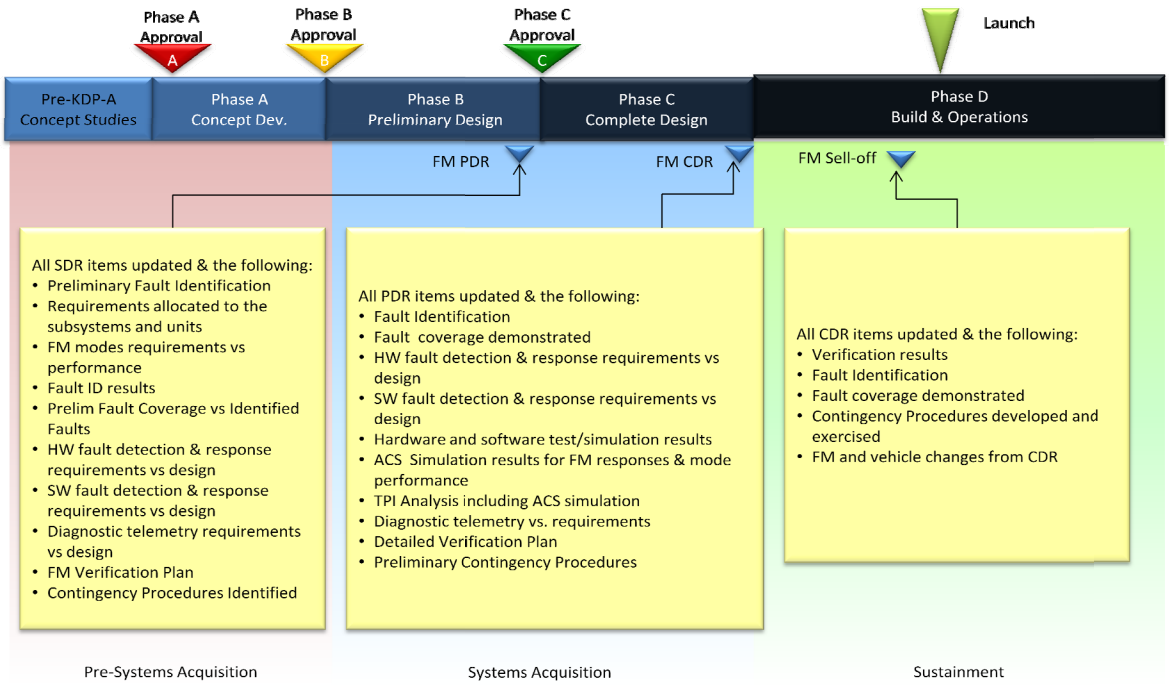


Figure 7. Fault management design reviews.

The primary purpose of the fault management design review is to provide the opportunity to determine if any errors or oversights were made in the requirements allocation or design process or in the design of the fault protection and fault management system. For fault management verification to be practical, the fault management system needs to be designed to be verifiable. The following features allow the design to be verifiable.

- Deterministic design
- Identification and separation of critical Fault Management functions from mission functions
- Simplicity
- De-coupling

A deterministic design limits the amount of needed analysis and testing. Simplicity enables the designer to predict the behavior of the system and limit the opportunity for error. De-coupling limits the amount of needed analysis and testing and reduces the chance for negative interactions between fault detection

monitors and responses. The reviewers should be able to look at the details of the information presented to assess not only whether the design will meet the stated requirements, but whether or not the design will protect the space vehicle should the identified faults actually occur.

4.1.2.2.2.1 Fault Management Preliminary Design Review (PDR)

The following items should be completed and demonstrated at the fault management PDR.

- Updated fault management CONOPs
- Preliminary fault identification completed and results presented
- Preliminary FMEA findings incorporated into the fault management requirements
- Fault coverage demonstrated (branch termination analysis if fault trees are being used)
- Fault management requirements allocated to the subsystems and units—includes hardware requirements, software requirements, and diagnostic telemetry requirements (both stored and real-time).
- Fault management modes updated and presented, including performance capability showing the design meets requirements
- Fault management architecture defined and presented
- Fault protection hardware fault detection and correction design compared to requirements (including what is swapped by what)
- Fault protection software fault detection and correction compared to requirements
- Fault management coverage for all vehicle activation (including deployments), mission operations, and modes
- Diagnostic telemetry compared to requirements
- Contingency procedures identified
- Fault management verification plan
- Schedule

4.1.2.2.2.2 Fault Management Critical Design Review (CDR)

The following items should be completed and demonstrated at the fault management CDR.

- Final fault management CONOPs documented and presented
- Fault identification updated with FMEA and results presented
- FMEA item to fault management trigger trace completed

- Fault coverage demonstrated (branch termination analysis if fault trees are being used)
- Fault management coverage for all vehicle activation (including deployments), mission operations, and modes
- Fault management requirements allocated to the subsystems and units—includes hardware requirements, software requirements, and diagnostic telemetry requirements.
- Fault management modes updated and presented, including performance capability showing design meets requirements (for each mode)
- Fault management architecture defined and presented
- Detailed fault protection hardware fault detection and correction design compared to requirement and design
- Detailed fault protection software fault detection and correction compared to requirements and design
- Diagnostic telemetry compared to requirements
- Preliminary thresholds, persistence, and interference (TPI) analysis results presented, including ACS simulation results
- Fault management hardware and software test/simulation results
- Detailed fault management verification plan presented
- Preliminary contingency procedures
- Lessons learned documented
- Schedule

4.1.2.3 Fault Management Sell-off

4.1.2.3.1 Verification Review:

The following items should be completed and demonstrated at the fault management verification review.

- Verification results—including TPI, ACS simulations, closed-loop testing results compared to design
- Fault identification, requirements, analysis updates for any changes past CDR
- Fault coverage analysis complete
- Fault scenario testing results summary

- Fault management effectiveness assessment results (Fault coverage analysis complete, scenario testing results summary)
- Contingency procedures developed and exercised
- Vehicle changes from CDR and the corresponding changes to fault management
- Update lessons learned

By sell-off, all of the hardware and software items have been tested or simulated using the appropriate test bed to show the fault management performs as designed [5].

4.1.2.3.2 Final as-built Review

The pre-ship review is the last chance for fault management to check that the fault management design has been implemented to an as-build design. This fault management review should show that the equipment as built is consistent with the requirements from fault management and that no new "design features" have been introduced post-CDR by equipment rework due to integration and test findings. The pre-ship review should address that the fault mitigation requirements have been implemented as intended, the fault protection is consistent with the as-built equipment failure modes, and that the fault tolerance requirement is met as a result of the mitigation and protection.

It is recognized that on many first-time space vehicle designs there are changes to the space vehicle after the fault management verification review has been held. Changes to the space vehicle after CDR usually have some impact to the fault management implementation. A final as-built review should be conducted to capture changes to the space vehicle design and their impacts to the fault management implementation to ensure a space vehicle design change has not introduced a flaw in the fault management system. It is expected that the FMEAs that support the fault management system have been updated when space vehicle design changes are made.

- Delta Verification results—including TPI, ACS simulations, closed loop testing results compared to design
- Delta fault identification, requirements, analysis updates for any changes past CDR
- Delta fault coverage analysis
- Delta fault scenario testing results
- Delta fault management effectiveness assessment results (Fault coverage analysis complete, scenario testing results summary)
- Delta contingency procedures developed and exercised
- Vehicle changes after the fault management verification review and the corresponding changes to fault management
- Update lessons learned

4.1.3 Fault Protection and Fault Management Guidelines

The guidelines provided below define the fundamental requirements for sound and effective fault management systems. The guidelines are grouped functionally and are ordered in a manner similar to how they need to be defined during a program.

4.1.3.1 Importance of Mission Assurance

As stated earlier, fault management is not suitable for addressing defects in the space vehicle design, software, or assembly. Redundancy also does not provide protection for common cause failures. Any existing design flaw will be present in both copies of the unit. Therefore sound mission assurance practices and standards should be employed to minimize the risk of common cause errors resulting from defects in the design, software, workmanship, or use of parts and materials inappropriate for the space environment.

Mission assurance requirements should include, but is not limited to, the following items:

Space Radiation Survivability Requirements such as single event effects (SEE) damage prevention, and allowable SEE upset rates requirement either ground or fault management intervention.
Parts, Materials, and Process Control Requirements
Safety Requirements
Reliability Requirements such as single-point failures (SPFs) elimination and mitigation plans and controls for retained single-point failures.
Space Environment Survivability Requirements such as proper coverings, plugs or caps for all test access ports to ensure environment effects do not propagate to internal components causing damage, electro-static discharge (ESD) protection, EMI/EMC protection, and micrometeoroid effects protection.
Launch Environment Survivability Requirements such as margins of safety.

Examples:
In the late 1990s, at least four commercial satellites had problems with their spacecraft control processors SCPs, reportedly because whiskers grew on the relays and caused the power-supply fuses to blow. In three cases, both the primary and the redundant SCPs failed, and the satellites were lost [5].

Again in the late 1990s, three DOD programs incurred costly delays: one discovered tin whiskers in an atomic clock, the second found tin whiskers on ground lugs, and the third saw tin whiskers forming inside thin-film capacitors [5].

4.1.3.2 Space Vehicle Class Definition

It is recognized that all space vehicle do not require application of all of the requirements and guidelines contained in this document. Four classes of space vehicle are defined to aid in the application of these guidelines. The space vehicle class definition does not recognize cost as part of the definition. Effect to national security, risk, and required on-orbit life are the factors used to assess the space vehicleclass [22].
Class A

A national asset space vehicle. National asset space vehicle generally support the national security of the United States. Class A space vehicles are the lowest risk (all practical measures are taken to reduce risk), highest importance to national security, highest confidence of success space vehicle (the highest procurement and assurance standards are used) with required long on-orbit life for a given orbit. Class A is reserved for space vehicle for which mission failure results in (1) an unacceptable collection gap, an

unacceptable delay of a new capability, or an unacceptable reduction of capabilities, (2) a potential human safety hazard, or (3) a security breach. A class A space vehicle is an operational asset.

Class B
A national asset space vehicle. National asset space vehicle generally support the national security of the United States. Class B space vehicle are low risk (most practical measures are taken to reduce risk), high importance to national security or safety (such as a high impact weather satellite), high confidence of success space vehicle (the highest procurement and assurance standards are used) with required moderate to long on-orbit life for a given orbit and represent the best industry standards for high reliability, high quality, and long design life space vehicle. A Class B space vehicle may be an operational asset.

Class C
A demonstration space vehicle. Demonstration space vehicle generally utilize new technologies to demonstrate new space vehicle capabilities. Demonstration space vehicle are moderate risk, some application to national security, moderate confidence of success space vehicle with required short to moderate on-orbit life for a given orbit. A Class C space vehicle is not an operational asset.

Class D
A proof-of-concept or fast-turnaround space vehicle. Proof-of-concept and fast-turnaround space vehicle generally are used to test new concepts or provide a high-risk immediate capability in space. Class D space vehicle are high risk, little to no application to national security, low confidence of success space vehicles with required short on-orbit life for a given orbit. A Class D space vehicle is not an operational asset.

The fault management guidelines for each class of space vehicle can be found in section 7.

4.1.3.3 Fault Management Mode Guidelines

Space vehicle fault management systems rely on modes which are developed to protect the vehicle from different types of failures.

Contingency Modes—Intermediate “safe” modes used to place the space vehicle into a benign operating state to establish a safe condition.

Safe Mode—A reduced functional fail-safe space vehicle mode in which a balanced power-safe, thermal safe, and communications-safe state is maintained allowing ample time for ground operations to recover to an operational space vehicle mode.

Low Power Response—A vehicle response used to reduce power loads designed to protect the vehicle in the event of a short, bus overload, or undervoltage condition.

The following section provides the recommended fault management mode requirements necessary for protecting the space vehicle. Table 7-1 details the fault management mode requirements applicability by space vehicle class.

4.1.3.3.1 Contingency Mode

Fault management may provide for a general purpose contingency modes which provide the following:

- Positive power/thermal balance

- Uplink communications with positive link margins
- Downlink communications with positive link margins
- Limited mission operations with protection for environmentally sensitive hardware

Rationale: Contingency modes can enhance space vehicle controllability and anomaly recovery time by employing intermediate reduced capability operating modes. Specific contingency modes implementations vary with vehicle design. Implementation details are provided in section 5.4.4.4.1 (Contingency Modes). Additional contingency modes may also be used, to facilitate ground recovery for less severe faults. These additional contingency modes can include a cruise mode for agile space vehicles, sun-point modes, and loss-of-communication mode.

4.1.3.3.2 Safe-Mode

Fault management should provide for a general-purpose, actively-controlled survival mode which provides the following:

- Positive power/thermal balance state
- Operationally benign state with protection for environmentally sensitive hardware
- Attitude control with robust stability margins using electrically independent and sufficient functionally independent sensors and actuators, such that primary and redundant attitude control hardware are not both rendered ineffective by the space vehicle state or environment.
- Electrically independent processors and data input/output paths for autonomous control functions.
- Uplink communications with positive link margins
- Downlink communications with positive link margins

Rationale: Specific safe mode implementations vary with vehicle design. Implementation details and examples are provided in section 5.4.4.4.2 (Attitude Control Safe Mode). Regardless of implementation, the general purpose actively controlled (safe) mode should use sufficiently independent attitude reference sensors and attitude control actuators. It is not always possible to correctly distinguish between attitude sensor, input/output, or actuator faults. Also, the space vehicle operating environment (radiation, magnetic storms, etc.) can render some attitude reference sensors ineffective. Likewise, safe mode attitude control actuators should have sufficient control authority to correct any anomalous space vehicle angular momentum states. Therefore, it is important that the general-purpose, actively-controlled safe mode provide the capability to configure to hardware and data paths not previously in use to provide fault containment and to prevent the fault from persisting in the safe made. The transition to the final safe mode state may be incremental or single step.

4.1.3.3.2.1 Safe-Mode Duration

The safe mode should be designed to keep the vehicle in the safe mode condition defined in paragraph 4.1.3.3.2 for a specified duration and specified number of entries.

Rationale: Ground should be allowed at least two weeks to be able to respond to a vehicle anomaly [5]. Many programs require longer safe state durations with 60 to 90 days being typical durations. Ideally the vehicle should be able to remain in the general-purpose, safe state indefinitely. However, the duration and number of entries are bounded for verification purposes. The number of entries should account for false alarms and other non-hardware failures. It is important to either not use or minimize propellant consumption in this state.

4.1.3.3.3 Low Power Response

The space vehicle should provide a deterministic low power response, independent of flight software which provides the following:

- Deterministic space vehicle state
- Power subsystem load shed, with only essential loads remaining (enabled)
- Independent battery charge control
- Independent critical telemetry
- Command capability needed for recovery
- Thermostatically controlled survival heaters to be enable by ground command

Rationale: A low power response is needed to protect the space vehicle in the event of a short, bus overload, or loss of power [1]. The goal of the low power response is to maximize the time the power system can support the load (minimize the load) to allow time to troubleshoot the issue. Attitude control does not necessarily need to be provided. Passive attitude control techniques, such as flat spin (for spinning space vehicle) or gravity gradient can be employed to provide a power-positive state for essential loads. For vehicles with constant ground contact, manual ground control can be used to control attitude.

Example: NOAA(I)
Abstract: Operations were normal for twelve days after launch until a short circuit in the battery-charge assembly resulted in a solar panel recharge failure and subsequent battery depletion. The most likely cause of the short was a screw on the assembly's heat shield which protruded too far into the assembly and made contact with a radiator plate. All attempts to contact or command the spacecraft since the power failure have been unsuccessful.

Symptom: Although solar array output indicated 31 amps, all three batteries were discharging and indicated low voltage. Twelve days after reaching orbit NASA engineers received indications battery charging stopped. Voltages and currents on the batteries steadily declined on subsequent passes of the satellite until contact was lost.

Cause: The probable cause of the failure was a short circuit in a battery charge assembly that prevented the solar array from powering the spacecraft. The short was most likely from a (31 mm) screw extending too far below a heat dissipating aluminum plate (at solar array potential). The screw end penetrated the insulation over time and made contact with a radiator plate (at battery potential) causing the short circuit. According to the Failure Review Board, the design of the battery charge assembly was unforgiving and could not be checked once it was assembled because the heat sink could not be removed or easily X-rayed. The design also required meticulous construction which presented many opportunities for a short

circuit. The situation was not helped when two scientific instruments were added, causing the system to run hotter. The assemblies were not re-qualified at the higher temperature.

Lessons Learned: (1) A failure-modes analysis on all spacecraft systems should have identified all potential single-point failures which would seriously impair function of the spacecraft. (2) During manufacturing, integration, and test the box was handled routinely rather than receiving the special handling it required, and new thermal requirements stressed an already fragile box and heat sink system [6].

4.1.3.3.4 Dead Bus Recovery Capability

The capability to recover to mission operation from a dead or low voltage battery condition should be provided.

Rationale: It is generally understood that recovery from a dead space vehicle battery cannot be guaranteed. However, space vehicle should be capable of being recovered provided there is a sufficient amount of remaining functional hardware available to do so. Specifically, this capability includes recovery from battery conditions where the primary bus voltage of a space vehicle collapses to any voltage below the minimum operational bus voltage, including zero, for any time interval exceeding normal or survival bus voltage limits. To maximize the probability of recovery to normal operations, additional autonomous actions are needed prior to, or during, bus voltage collapse [23].

Examples:
Olympus: The Olympus satellite suffered an anomaly on 29 May 1991 that left the satellite tumbling. Soon the batteries discharged, the satellite cooled down, and the propellant froze. Worse, autonomous thruster firings during the tumble had introduced a Delta-V of 15 meters per second, which had nudged the satellite off its orbital slot. The only transmission from the satellite was an unmodulated carrier. At times, the solar arrays were able to power the transmitter. Using various assets, and the power variation when the transmitters were powered, the controllers were able to determine that the power situation would improve by mid-September. By 19 June, the bus voltage was sufficient to allow the satellite to receive ground commands. By the end of the month, heaters had been activated to warm the nickel-hydrogen battery, to enable it to be recharged, but there was not yet sufficient power to run the heaters and the charger simultaneously. In July, the propellant was thawed. The final steps began on 26 July, with the thawing of the thrusters. Once the spin rate was reduced to 2 degrees per second, the satellite was able to go autonomously into Sun-acquisition, and require an Earth-pointing orientation. A burn on 13 August halted the longitude drift in order to resume the original geostationary point. The recovery took thousands of telecommands, as well as extensive rehearsal using a satellite simulator. By 19 August, the recommissioning of the payload began and operations then resumed [3].

SOHO: On 25 June 1998, two months into its extended mission, the satellite was lost as a result of maintenance activity that left Gyro B in the high-gain setting. Gyro A had previously been turned off to help preserve operating life. The software did not recognize that Gyro A was off, which led the software to perceive that the spacecraft was in a roll, which it tried in vain to correct by firing the thrusters. On the basis of the discrepancy between gyros A and B, the operators made a diagnosis that Gyro B was faulty, and switched if off. SOHO tried to recover using the inactive Gyro A and it began to wobble. Soon it was spinning at 7 revolutions per minute and coning about its X-axis at 60 degrees and it adopted a third safe mode. Contact was lost within 5 minutes. On 3 August SOHO was coaxed into activating its transmitter. Over the next few days the telemetry was activated. The battery charging strategy was changed and the power situation began to improve. Regaining control would require the propulsion system to be thawed, because the spin rate was too rapid for the momentum wheels to counter. One communications session that lasted some 100 minutes established that the batteries were charging.

As the power was restored, the spacecraft was able to be thawed out. Despite some of its instruments having been chilled down to -122°C, the spacecraft was able to continue operating [3].

4.1.3.3.5 Fully Independent Safe Mode

Fault management may provide a fully-independent, actively-controlled safe mode as needed to meet additional mission required fault management requirements.

Rationale: A fully independent safe mode uses a dissimilar processor, attitude sensors, and actuators from the normal operating space vehicle modes. Such a mode requires additional hardware to implement, and thus increases the weight and volume of the space vehicle bus. Inclusion of such a mode also adds complexity into the space vehicle and fault management design. The additional complexity may introduce single-point faults, which may be difficult to identify. The independent safe mode would require different design and construction to provide protection against common cause faults. If this mode is used during space vehicle activation, it should also be single-fault tolerant.

4.1.3.3.6 Manual Recovery

Recovery from safe mode or low power response should be under ground control.

Rationale: The modes are intended to provide a safe state to allow ground recovery. The general-purpose safe mode is usually entered for attitude control anomalies for which the source of the fault cannot be autonomously isolated. The low-voltage response would be entered in the event of loss of attitude control or an unexpected load on the bus. Fault management cannot readily address common failures, nor can it be proven that all possible faults are protected. Therefore, recovery from these modes should be performed under ground control, to allow ground-fault isolation as needed to perform the recovery.

There are some instances where limited autonomous recovery may be desired. Autonomous fault recovery should then specify these conditions and the allowed recovery actions.

4.1.3.3.7 Fault Management Mode Recovery

The fault management system should provide recovery time requirements to the subsystems to aid in returning the space vehicle to operation within the required recovery time.

Rationale: It is recommended to design the space vehicle to be recoverable within the required recovery time. These times are needed to drive the design so that recovery to the operation state can be done in a reasonable amount of time or as defined by mission requirements. The recovery time is starts after ground operations have decided to return to service.

Example: During the design of a space vehicle payload system, it was realized that the implementation of the payload computer required extensive data tables to provide default settings in the event of an on orbit swap to meet the mission downtime requirement. The issue was that the default tables were very large and would have required 4.5 months of dedicated contact time to upload the tables. The other option was to provide storage on board to provide a source for the default tables. The decision was made to design in a new memory (non-volatile) board to store the default tables to allow the space vehicle to meet its mission downtime requirement.

4.1.3.3.8 System Disposal Guidelines

It is necessary to be able to properly dispose any failed or decommissioned space vehicle.

Space vehicles for which an uncontrolled re-entry poses a human safety hazard should have independent (design and construction) means to perform the de-orbit maneuvers. Safety required independent system disposal should be single fault tolerant.

4.1.3.3.8.1 System Disposal

The space vehicle should be capable of performing orbit-specific vehicle disposal operations.

4.1.3.3.8.2 System Disposal Fault Tolerance

No single fault should prevent the ability to dispose of the space vehicle.

4.1.3.3.8.3 Independent System Disposal

Space vehicles for which an uncontrolled re-entry poses a human safety hazard should have independent (design and construction) means to perform the de-orbit maneuvers.

4.1.3.3.8.4 Independent System Disposal Fault Tolerance

Safety required independent system disposal should be single fault tolerant.

4.1.3.4 Fault Management Requirements Derivation

Except during critical event periods, the primary purpose of an autonomous fault protection system is to place the space vehicle in a safe, commandable state which can be maintained for a reasonable period (typically two weeks) following a fault. During critical periods, the primary purpose of the fault protection system is to ensure the completion of the critical event [7]. In some mission applications, it may be desirable to operate through some faults to minimize mission downtime. However, the need to isolate the fault and maintain the vehicle in a safe operating condition should always take precedence.

The fault management systems engineer should be responsible for flowing down the system-specification fault management requirements to both the space and ground elements, and to the affected space vehicle subsystems and units (including payload).

The fault management systems engineer should participate with the subsystems engineers to define and develop specific fault detection monitors and fault responses to correct the effects of the identified faults. These detailed requirements include both hardware, and software-implemented fault management. Table 10 details the fault management requirements derivation guidelines applicability by space vehicle class.

4.1.3.4.1 Fault Mitigation Requirements Allocation

Fault mitigation requirements should be allocated to hardware to mitigate faults as defined in Section 4.1.3.4.4 (Fault Mitigation Design Guidelines) before defining the necessary fault protection requirements. These requirements should be flowed down to the various subsystems as applicable. There may be other sources for these requirements, such as launch vehicle and/or ground safety requirements as well. Once these requirements are in place, additional fault protection requirements are usually not needed. However, it is important for the fault management system designer to know and be aware of where this fault protection is applied. In some cases, it may be necessary for the hardware designer to alter the provided protection, and then autonomous fault protection may be needed. Fault trees are particularly useful for identifying where this protection is in place and showing where autonomous fault protection may be needed.

4.1.3.4.2 Fault Protection Requirements Allocations

After the hardware fault mitigation requirements have been identified, the autonomous fault protection requirements can be allocated.

Fault protection requirements should be allocated to the onboard elements of the system in accordance with the following principles [5]:

- *Space vehicle versus ground control.* Autonomous fault protection should be included onboard the space vehicle only if a response by Mission Operations is not feasible, practical, or timely enough to ensure the safe condition of the space vehicle. Otherwise, ground control should be responsible for fault response. For missions that have command or telemetry outages, onboard fault protection should be responsible all the time, not just between passes. In both cases, ground control is responsible for failure diagnosis and, if necessary, the reconfiguration of the space vehicle to nominal operations after the fault.
- *Protection against operator errors.* Known hazardous commanding (irreversible event) or known hazardous operating conditions, such as sun exposure, should be protected onboard.
- *Protection against space vehicle hardware and software design errors.* To simplify the development of fault protection, autonomous fault protection is usually not required to protect against space vehicle design errors, although it is not prohibited if practical. The practice of fault protection typically provides some limited space vehicle protection against design errors (e.g., thermal fault responses, controller instability, etc.).

All on-board, post lift-off, autonomous fault protection requirements can be allocated as either "subsystem internal" or "system" fault protection. Engineering elements which have been allocated fault protection responsibility should further derive and allocate the requirements and design for the associated detections, monitors, responses, and diagnostic data in compliance with project functional requirements. Where mission payloads include fault mitigation and/or fault protection in their design, designers must still ensure compliance with all space vehicle project fault protection requirements.

4.1.3.4.3 Fault Management Performance Guidelines

In addition to functional requirements, fault management will typically have performance requirements that must be satisfied. Generally, these are derived requirements that are space vehicle-architecture and design specific, but are generally good engineering practice or result in life-extending features built in to the space vehicle. Table 11 details the fault management, performance-derivation guidelines applicability by space vehicle class.

4.1.3.4.3.1 Fault Tolerance

Fault tolerance of the space vehicle can be a very time-consuming and expensive requirement to implement. This section contains several different levels of fault tolerance. It is strongly recommended that the customer and contractor have a very frank discussion on the level of fault tolerance desired, including the interpretation and implementations implications of the requirement from all parties.

Along with the reliability requirements, the fault tolerance requirements define the amount of redundancy and cross-strapping provided by the space vehicle design.

4.1.3.4.3.1.1 Single-Fault Tolerance

The space vehicle should be designed to be single-fault tolerant, during all activation and mission modes, allowing recovery to full mission operations after a single fault. The common interpretation of single-fault tolerance is that the space vehicle is able to tolerate a single fault (system wide) only.

Rationale: Faults are assumed to occur one at a time, with multiple simultaneous faults occurring to be extremely low probability. The space vehicle still must comply with any additional safety requirements for additional fault tolerance to support ground safety and launch vehicle safety requirements.

4.1.3.4.3.1.2 Enhanced Single-Fault Tolerance

The space vehicle should be designed to be enhanced single-fault tolerant, during all activation and mission modes, allowing recovery to full mission operations after a single fault. The common interpretation of enhanced single-fault tolerance is the space vehicle is able to tolerate a single fault *at a time*, followed by isolation and recovery, followed by another fault some time later. As long as no two of the same *function* is affected, the vehicle should still be safe (although the space vehicle may be in a degraded condition). Enhanced single-fault tolerance is also referred to as graceful degradation.

Rationale: Enhanced single-fault tolerance for subsequent failures should be applied whenever practical. At a minimum, fault protection should be designed with the assumption that only one fault occurs at a time, and that a subsequent fault will occur no earlier than the isolation/recovery activity for the first fault. As a goal, fault protection should be capable of recovering from multiple successive or coincident faults provided that the faults and associated fault algorithms are independent [5].

4.1.3.4.3.1.3 Multiple Fault Tolerance

The space vehicle can be designed to be multiple-fault tolerant for a small number of critical functions. Generally, this applies to critical items such as insertions burns, battery protection, and man-rated systems. This is sometimes referred to a dual-fault tolerance.

4.1.3.4.3.2 Fault Coverage

Autonomous fault protection should provide for fault correction of random hardware faults. Fault protection may provide fault correction for hazardous constraint violations, hazardous vehicle conditions, or SEU effects that cannot be mitigated by other means.

Rationale: Fault protection is most suitable for single, random, hardware fault correction. It can be utilized to provide hazardous constraint protection and should protect against anomalous conditions that can cause damage to non-redundant hardware. An example of a hazardous constraint would be line-of-sight constraints for sensitive hardware to protect against sun exposure.

4.1.3.4.3.3 Safe Mode Entry and Exit Fault Tolerance

No single fault should prevent successful ground commanded or autonomous entry to and exit from safe mode.

Rationale: A single failure should not prevent entry into this mode or subsequent exit from this mode to allow mission operations to continue should the mode be entered underground command or autonomously in the event of a false alarm or operator error.

4.1.3.4.3.4 Independent Command and Telemetry

The space vehicle should not have any single failure that can affect the command path and the telemetry path at the same time.

Rationale: It is a desired feature of the space vehicle to have command and telemetry independent so that only command or telemetry is lost due to a single failure, but not both.

4.1.3.4.4 Fault Mitigation Design Guidelines

Fault management has limited ability to protect against common cause failures. To maximize the ability of fault management to protect the space vehicle, it is prudent to include features into the space vehicle design that can either eliminate or mitigate potential mission ending failures and simplify the fault protection design and implementation (increase the fault management's probability of success). Table 12 details the fault mitigation design guidelines applicability by space vehicle class.

4.1.3.4.4.1 Fused Power Considerations

If the primary power bus on the space vehicle is a non-redundant power bus, primary power to units should be fused.

For loads above a mission-specific maximum allowed fault current, fault management needs to impose requirements to either have a second independent means of turning off loads exceeding a specified maximum allowed faulted current, or imposing a maximum fault current, on each load exceeding the maximum allowed current. The maximum allowed load value should be within the thermal and energy balance margins for the space vehicle.

Rationale: The fusing of primary power is to protect the spacecraft power bus the distribution wiring from faults internal to each load receiving power from the bus. Because the primary spacecraft bus is not redundant, fuses (or circuit breakers) are needed to disconnect a faulted load. Fuses are typically de-rated to about 50% of the peak steady state current consumed by the load, and can carry up to 200% of their rating. This creates a so-called "smart fault" zone between the maximum normal load current and the current required to blow the fuse. For example, a load that draws a maximum steady state current of 7 amps requires a fuse rated to at least 14 amps. Since there are no 14 amp fuses a 15 amp fuse would be selected. A 15 amp fuse can carry up to 30 amps without blowing. Fault management needs to be concerned if a faulted load draws current in the "smart fault" zone between the normal 7 amp and the 30 amps required to blow the fuse. For example, assume a "smart fault" in our load draws 20 amps from a 30 volt bus. 20 amps will not blow the 15 amp fuse, but it will dissipate 600 watts into the thermal subsystem and negatively affect energy balance. Such faults should be considered a credible fault in the detailed fault protection algorithm design and appropriate fault monitors and responses defined if the resulting "smart short" load is not within the space vehicle's thermal and energy balance margins.

4.1.3.4.4.2 Physical Separation

Primary and redundant functions should be physically separate.

Rationale: The most effective approach to achieving separation between prime and redundant functions is to physically separate prime and redundant circuits (separate units). If both prime and redundant circuits are housed within a unit, it is recommended to put these circuits on separate printed wiring boards (PWBs) with signals routed through different connectors.

When this approach is not possible because of design restraints, the guidelines in TOR-2008(8546)-7335 [8] should be followed. Internally redundant PWB assemblies for space applications have a unique set of requirements for physical separation of prime and redundant signals and power planes. The minimum recommended distances and geometries are derived from experimental and field failure data. The separation distances are unique as the consequence of a prime signal to a redundant signal short may occur resulting in a single-point failure [8].

Traces on the same PWB layer are subject to shorting due to conductive anodic filaments (CAF). The other area of consideration is prime and redundant signals and/or vias on adjacent PWB layers. Traces on adjacent PWB layers are also susceptible to shorting due to the CAF mechanism [8].

4.1.3.4.4.3 Assembly Error Mitigation

Designs in which an assembly error could result in a catastrophic failure should have features designed to prevent the error from occurring during assembly.

Rationale: Fault management is not designed to compensate for structural or mechanical failures (common cause design escape).

Example: On the Alexis space vehicle, a solar panel broke loose during launch, damaging the magnetometer and causing the satellite to tumble. Fortunately, the spacecraft was able to use Earthshine to trickle charge permitting ground controllers to regain control six weeks after launch. By using ground-based torquer control, most mission requirements were met.

Symptom: Onboard video showed that the +Y solar paddle had prematurely separated from the satellite. The magnetometer attached to the solar paddle was damaged, throwing the spacecraft off balance.

Cause: A bracket that had been damaged during ground testing broke loose during launch, leaving one of the satellite's four solar panels dangling by only its electrical cabling. Four solar array paddles were attached to the spacecraft with aluminum brackets and three brackets were stiffened with gussets. However, due to interference with surrounding components, a gusset used to provide stiffness to the fourth bracket was not included in the design of the +Y bracket hinge. During the random vibration tests, the overly flexible +Y bracket overloaded the release mechanism clevis at the other end, causing it to sustain significant damage. The release mechanism failed during launch, breaking the hinge bracket and leaving the paddle dangling by its electrical cabling.

Recovery: Video onboard the Pegasus launch vehicle revealed that recovery was possible despite the damage. Several weeks later, a downlink indicated that the satellite was able to charge up partially via Earthshine. By controlling the magnetic torquers from the ground, operators were able to determine the satellite's orientation and begin on-orbit operations three months after launch.

Corrective action: (1) Thoroughly understand all of the loads that act on the satellite system during ground testing, transportation, and launch. (2) All structural elements that provide a load path for supporting satellite components should be analyzed and the analyses should be independently reviewed. In addition, records of the analyses should be retained [6].

4.1.3.4.4.4 Fail-Safe Interface Designs

Cross-strapped interfaces should be designed with fail-safe circuitry to prevent fault propagation.

Rationale: Interface requirements need to account for off-nominal or transitory conditions that can result from a fault or single-event effect. Protection to prevent fault propagation is needed for cross-strapped designs so that a single fault cannot propagate to a working unit.

4.1.3.4.4.5 Hazardous Command Protection

No single command should permanently damage the vehicle or cause an irreversible event.

Rationale: Two-step commands provide protection so that no single fault or erroneous command can cause a catastrophic or non-reversible event to occur. The design should require two or more independent human errors, two or more failures (includes radiation event), or one failure plus one human error for catastrophic or non-reversible event to occur. No further protection is required once this mitigation approach is implemented and verified. Two-step commands are sometimes referred to as ARM/EXECUTE or ARM/FIRE command pairs

Example: Example hazardous events requiring two-step commands are ordnance firing for deployments, thruster activation, and non-reversible hardware reconfiguration.

4.1.3.4.4.6 Spurious Command Protection

No single fault should result in hazardous spurious commands.

Rationale: Faults should not cause spurious commands on the space vehicle to be sent. Ideally, hardware design should prevent spurious commands from occurring due to failures or during off-nominal operation. In this context, spurious commands includes any unintended change of state i.e. the output of a discrete during power up or down

Example: This is an example that resulted in the loss of a space vehicle on its first orbit. The aperture cover's design called for its pyrotechnic circuits "safed" prior to being sequentially "armed" and "fired."

A design feature in the controller chip invalidated all the programming circuits for a few milliseconds upon powering up. All outputs, including "ARM" and "FIRE," were momentarily asserted. The cover blew open prematurely; the cryogen escaped. The chip would manifest this start-up problem only after having been turned off for several hours. Although power cycled many times during component testing, it was never unpowered long enough to reveal the problem. The use of a slow, non-flight-like, power supply during unit testing masked the spurious output: during the transient period there was not enough voltage to close the arming relays. Later, anomalies repeatedly occurred during system testing. Unfortunately, because the pyrotechnic simulator was very sensitive, a load delay was fitted to the test equipment to filter out spurious triggers, unintentionally preventing the actual start-up glitch from being recorded. The warning signs were ignored.

At launch, the chip had been powered down for weeks. Not only did it go awry but, because power to the pyrotechnic box was applied via a fast relay, sufficient voltage had also built up to complete the arming circuit. The FIRE switch, commanded by the same controller and therefore not truly independent, set off as well, ending the mission. This controller chip had caused troubles before, prompting NASA to issue an application note. However, the contractor and the field engineer from the vendor did not know about it. " [We need] an information hotline, set up on an industry-wide, lessons-learned Web page," suggested the engineers later [6].

4.1.3.4.4.7 Command Lockouts

Commands should never be locked of any equipment essential to maintain control of the space vehicle or to dispose of the space vehicle. The ground should have the ability to override an onboard function and have full control over the vehicle.

Rationale: The ability to control the vehicle from the ground for critical activities (de-orbit operations, safe mode recovery) should be preserved. The term "control" here is intended that the ability to control the configuration of the key equipment to switch to alternate onboard or ground control is available. Some space vehicles incorporate a small command lockout window to allow the onboard, safe-hold, command sequence to execute uninterrupted by the ground. If a small command lockout window is used to allow the on-board safing command sequence to execute uninterrupted by the ground, then the command lockout period should be short (on the order of a few seconds) and deterministic and not have any failure modes that could cause a failure in the lockout state.

4.1.3.4.4.8 Conditional Commands

The use of conditional commands, i.e. commands that rely upon the execution of a previous command, should require the verification of all preceding commands necessary for the final command to be executed.

Rationale: The use of conditional commands is very useful to the operation and safing of a space vehicle. The use of "arm" then "fire" commands are necessary when commanding items (like ordnance) where a single command could have disastrous results. For these occasions, the separate execution and verification of the commands is required to ensure the space vehicle is in the proper configuration prior to the final command.

Example: The ground crew routinely sent commands in two parts: "clear register" followed by "execution," but the command checking process would not preclude "execution" in case the "clear register" command was not carried out. A command was issued when the satellite was at a low elevation off the horizon. The "clear" instruction did not go through, but the "execution" did, causing the flight computer to freeze. Numerous stored command sequences backed up. During subsequent contact with the ground station, the operators reset the interrupt controller. Right away, all pending commands ran. The telescope aperture door and the calibration source motor, intended for operation with a 40-second pause in between, turned at the same time. The telescope controller, not designed to accommodate the simultaneous mechanism operation, blew its fuse. Fortunately, the aperture door automatically sprang open to allow the mission to continue, but on-board calibration was no longer possible [6].

Example: The MTI satellite's telescope calibration (TCAL) control unit failed, thereby disabling the onboard calibration system. The loss of TCAL—which controlled the telescope heaters and mechanisms, including the aperture door, calibration wheel, and focus mechanism—left the telescope slightly out of focus, but fortunately not terminating the mission because the onboard, fail-safe design automatically opened the aperture door and swung the calibration wheel away, preventing it from interfering with telescope operation.

Symptom
During the first ground contact in November 2000, ground station telemetry reported that the Telescope and calibration (TCAL's) control unit's +5 volt supply was not functioning and that the telescope door was open.

Cause
The root cause of the failure is an unsafe commanding process, coupled with inadequate fuse sizing. An uplink issued when the satellite was at a low elevation off the horizon did not completely go through, causing the flight computer to seize and numerous stored command sequences to back up. When the operators reset the interrupt controller, all pending commands ran. The telescope aperture door and the calibration source motor, intended for operation with a 40-second pause in between, turned at the same time, blowing a fuse that, in retrospect, should have been sized larger.

Recovery
Upon power loss the telescope door sprang open, permitting data collection to continue, using calibration sequences developed prior to the failure.

Lessons Learned
(1) Conditional commands (execution of an instruction contingent upon another) must first verify the completion of the preceding command. (2) If multiple commands can cause a mechanical or electrical conflict, code in a prevention block (i.e. an exclusive OR). (3) Make sure flight computers are restarted in a known mode with only appropriate commands in the queue—always clear pending commands first. (4) Double-check if a fuse should be installed, and carefully analyze fault scenarios to size fuses [6].

4.1.3.4.4.9 Single Function Commands

The space vehicle should implement commands that result in one intended function for a given bit sequence

Rationale: Each bit sequence should result in one and only one action to avoid having single bit sequences have more than one possible action. Multiple commands for a given bit sequence are confusing (as they are context sensitive) and can result in the wrong action occurring on the space vehicle. The use of “toggle” commands, that is, a single command to turn a function on and off, depending upon the previous state, should not be used. The response to a command should be deterministic based upon the command sent. Deterministic commanding is needed in the event if commands must be sent “in the blind” (telemetry not available).

Example: An example of a command that does not meet the single function requirements would be a command of ‘11111111” to turn a processor on, and if the same command was sent again, turn the processor off.

4.1.3.4.5 Fault Identification

Faults should be identified through a systematic analysis process encompassing detailed analysis of space vehicle hardware and software, and the operating environment.

Rationale: The combined Fault Tree Analysis (FTA) / Failure Modes and Effects Analysis (FMEA) approach is recommended. Historical failure modes, including lessons learned from other space vehicles should be considered and used. Methods for performing fault identification are discussed in section 5.2 (Fault Identification Methods).

The combined FTA/FMEA approach should be used to assist in the system design, to identify what faults should be protected by hardware fault mitigation techniques, to identify autonomous fault protection requirements and to show that an adequate level of fault detection coverage has been achieved. The method is widely used and accepted, though it is labor intensive and prone to error. It is important to

perform both types of analysis. Neither analysis method on its own is sufficient. The effects of common mode failures can be incorporated into the FTA, though the failure probabilities may not be known [1].

Generally the FTA should be used as the primary means to identify requirements and the FMEA as the primary input to the design of the monitor algorithms to meet these requirements. Both types of analysis should be used to optimize the design of monitor algorithms, just as both types of analysis should be used to identify the fault detection requirements. In this way, monitor algorithms should be designed to detect real failure modes that might cause, or contribute to, some hazardous behavior of the system [1].

Although the FTA and FMEA are useful in defining the detection requirements and designing the fault detection algorithms, there is no single methodology available to the system designer that fully supports this task. Consequently schemes for fault detection and isolation are also based on a collection of different methods, experience, and engineering judgment.

Fault detection algorithms based only on the FTA tend not to succeed. The most common problem encountered is that monitor thresholds are set unnecessarily tight resulting in false positives. While thresholds biased towards false positives can result in more false alarms, they do not mask fault conditions in the long run. Similarly, fault detection algorithms based only on the FMEA rarely succeed. The main problem encountered here is the definition of monitoring algorithms without proper consideration of the need to detect the failure. Again the monitoring algorithms may be unnecessarily stringent [1].

As discussed above, it cannot be proven that the combined FTA/FMEA approach will identify all possible faults. Therefore other techniques, such as passive survival modes or fully independent safe modes (sensors, controllers, and actuators) should be considered in addition to autonomous fault protection to help safe guard against escapes in the FTA/FMEA fault identification method. Table 13 details the fault management fault, identification guidelines applicability by space vehicle class.

4.1.3.4.6 Space Vehicle Safety Device Guidelines

Safety devices should be utilized to protect non-redundant hardware from permanent damage or limited life resources from unintended depletion. These safety devices must be independent from any fault protection algorithms to be effective. Independence requires the use of different design and different parts. The safety devices described below are intended to protect an orbiting space vehicle. Ground or launch safety may require additional safety devices. Table 14 details the space vehicle safety device requirements applicability by space vehicle class.

4.1.3.4.6.1 Battery Over-Charge or Over-Temperature Protection

Independent protection should be provided to prevent either a hazardous condition or permanent damage to the batteries resulting from over-charge. Space vehicle battery over-charge or over-temperature conditions that can result in battery explosions or permanent damage should be dual-fault tolerant.

Rationale: Generally, battery charge and battery temperature are managed by the flight software. Independent battery over charge and battery over temperature protection should be implemented to avoid damaging the batteries should the flight software experience an anomaly (software logic error, coding error, hardware anomaly, loss of attitude control) prevents the battery charge or temperature from being managed to avoid damage to the batteries. Dual-fault tolerant protection is needed to protect batteries from conditions that can result in an explosion. Not only does a battery explosion threaten the effected space vehicle, the resulting debris can threaten other space vehicle as well

Example:
Abstract
Contact with ABRIXAS was lost due to problems with the battery.

Cause
During the second contact, three hours after launch, a high battery temperature was noted. On 29 April there was a sudden change in the battery voltages. Data show that either one cell was damaged or, more probably, the connection between the battery and the rest of the power system was interrupted (it is unknown here whether they mean the single battery or the string of 11, nor do they know whether 'power system' means solar cells). For the following two days, ABRIXAS ran off of the power of its 'starter battery' which is not chargeable. On the night of 30 April, contact with ABRIXAS was lost.

Recovery
Scientists are hoping that the six-day stretch of sunlight will act as 'therapy' for the batteries, but this idid not work [6].

Example:
NOAA 8: In June 1984 a battery charge regulator failed. The controllers treated the battery with care, manually setting a low charge rate, and the satellite resumed service. On 30 December 1985, while the satellite was out of contact, the power supply for the attitude control system's clock oscillator failed momentarily, causing the computer to adopt a safe mode that included a default mid-range charge setting for battery number 1. Nine hours later, the telemetry indicated that this battery was inoperative, and the satellite was tumbling. On 3 January, 1986, NORAD warned that it was tracing several fragments close to the satellite. Nevertheless, contact was regained four days later. The telemetry history indicated that five hours after the battery charging had been set to the mid-range value the battery had overcharged, undergone a thermal runaway, and exploded, possibly as a result of two of its cells having been weakened by a previous anomaly. The explosion had blown off the thermal blankets." [3]

Example:
SUNSAT (Stellenbosch University Satellite): The 64-kilogram microsatellite hitched a ride on a Delta II on 23 February 1999, and was released into a 400 by 840 kilometer polar orbit. For the first year, the plane of the satellite's orbit enabled it to cool down while it was in the Earth's shadow. However, as the orbital plane precessed to become normal to the Sun direction, the satellite was in continuous illumination for five months. The microsatellite had only modest margins on its thermal performance, and a simple power system. Although the satellite was reoriented in an attempt to mitigate the situation, it suffered high temperatures and overcharged its batteries. Thereafter, the battery voltage tended to rapidly decrease under load, and while it was in shadow it fell to the point where the processors reset. This suggested that the nickel-cadmium batteries had degraded [3].

4.1.3.4.6.2 Battery Depletion Protection (Li-Ion)

Independent protection should be provided to prevent depletion of the batteries that would result in an unrecoverable condition.

Rationale: Lithium-ion (Li-ion) batteries can be damaged if they experience an undervoltage condition (unlike nickle hydrogen or nickle-cadmium batteries). If the voltage on a Li-Ion cell voltage becomes less than the cell critical voltage, permanent damage can occur. If an attitude control, flight software, or other anomaly can result in an undervoltage condition to a Li-ion battery, there should be independent protection (from flight software) of the battery [9].

4.1.3.4.6.3 Structural Limits Protection

Independent protection should be provided to detect and protect against excessive space vehicle or mechanism (wheel) rates using safety shut-offs.

Rationale: Space vehicle and unit designs should include independent protection against exceeding structural limits. Hardware failures can cause actuators to fail on, causing a potentially hazardous condition.

Examples: Momentum or reaction-wheel overspeed protection. CMG gimbal overrate protection. Thruster time-outs. Pressure-vessel burst pressure margins.

4.1.3.4.6.4 Propellant Depletion Protection

Independent protection should be provided to prevent an anomaly from causing the depletion of the propellant. Generally flight software is responsible for firing thrusters to control attitude. To provide protection from flight software logic errors from expending all of the propellant autonomously, an independent valve firing, propellant usage protection should be implemented.

Rationale:
Uncontrolled propellant (expandable) usage can quickly end a space vehicle's life. Propellant is a life-limited resource that is used for attitude control and orbit maintenance. Protection should be provided to prevent inadvertent loss of large amounts of propellant that can result from erroneous operation. The capability of the space vehicle to be recovered from such an anomaly requires dead-bus recovery capability, in that termination of propellant consumption can result in a loss of attitude control and eventual loss of power.

Example:
The Problem: A deep-space mission ended prematurely after excessive thruster firings depleted its fuel.

The Cause: This spacecraft was developed by a highly-motivated group operating under a rigid cost cap and tight schedule. Flying just 22 months after being funded, it successfully circled the moon and demonstrated many technologies.

Soon afterward, however, a maneuver triggered a numeric overflow in the processor, causing it to erroneously fire its thrusters and freeze. A "watchdog timer" algorithm should have stopped the thrusters from continuously firing, but did not execute because the computer had already crashed. By the time ground operators regained control, all the fuel was gone.

A hard-wired timer, which would have stopped thruster firing, was not implemented due to the tight schedule. Time pressure also prevented the software from being fully tested, and many changes had to be uploaded as faults were discovered.

The overflow error had occurred thousand of times (without causing malfunctions) because the project had to settle for an inadequate but available processor. Software changes had been written to correct the problem, but the overstretched staff could not handle operations, anomaly analysis, and software repair at the same time, and the change was not loaded.

Four years later, another interplanetary probe encountered a similar anomaly. Fortunately, engineers learned the lessons from the previous incident; the precautions they took allowed them to successfully complete the mission.

Lessons Learned

Apply independent fault protection for critical software functions.

Implement exception handling to protect the flight processor from aborts due to data-handling errors.

Do not cut corners in testing critical flight software [6].

Example:
Abstract
During maneuvers to circularize DFH 3-1's orbit to geosynchronous earth orbit (GEO), anomalies caused expenditure of the complete lifetime budget of attitude-control propellant.
On station in GEO orbit, the vehicle was unable to maintain operational attitude control due to a lack of propellant, and was abandoned.

Symptom
DFH 3-1 was launched into geosynchronous transfer orbit (GTO).
During this orbit transfer from GTO to GEO several different anomalies were reported, including
(a) "AKM malfunction(s)",
(b) an "apparent leak" in the propellant piping of the attitude-control system causing excessive loss of propellant and perturbations to the vehicles attitude,
(c) "navigation and attitude control system problems, " and
(d) AKM burn(s) that were prematurely terminated due to "a navigation malfunction" and "attitude control systems" problems.

These anomalies required deviations to the planned AKM burn duration and sequence that temporarily left DFH 3-1 stranded in a quasi-geostationary transfer orbit. As a result of several anomalies, DFH 3-1 vehicle instabilities occurred during the orbit-transfer process, requiring excessive and unplanned amounts of attitude-control propellant to be expended during AKM firings.

Cause
Upon reaching its GEO-operational station, the full lifetime budget of attitude-control propellant had been prematurely used up to maintain vehicle stability during the AKM firings. So much propellant was used during this effort to stabilize the space vehicle during its struggle into GEO, that there was not enough propellant to complete the final drift maneuvers or to maintain operational station keeping once the satellite was on station.

Recovery
Without any remaining propellant, DFH 3-1 was abandoned because it was unable to maintain attitude control for the operational phase of its life. Due to these anomalous orbit transfer maneuvers, DFH 3-1 final orbit station was 132 degrees east, instead of the intended 125 degrees east. Vehicle abandoned at 35,181 x 35,993 km, 0.26 deg (a GEO drift orbit) after control problems exhausted propellant [6].

Example:
Clementine: Clementine was launched from Vandenberg on 25 January 1994 by a refurbished Titan II, entered lunar orbit on 19 February, functioned flawlessly, and departed on 3 March 1994 to rendezvous

with its primary target in August. Unfortunately, four days later, during a 20-minute telemetry hiatus while preparing for a maneuver, a flaw in the software of the autonomous computer fired the attitude control thrusters for 11 minutes, exhausting the supply of propellant and leaving the spacecraft spinning at 80 revolutions per minute, which forced the abandonment of the mission [3].

4.1.3.4.6.5 Closable Shields

A closable shield door may be provided to protect sensitive hardware from exposure to space environmental hazards.

Rationale: Some hardware elements can be damage from prolonged exposure to the space environment (sun, cold, excessive radiation, debris). Closable shields (doors) are typically used to provide protection for sensitive hardware.

4.1.3.4.7 Command and Control Fault Protection Guidelines

The effects of operator error can be mitigated by use of the features described below.
Table 7-7 details the command and control fault protection guidelines applicability by space vehicle class.

4.1.3.4.7.1 Critical Commands

No single ground command should be able to place the vehicle in a potentially hazardous state without some override mechanism (ground override is acceptable).

Rationale: A critical command is any command that changes the configuration of the space vehicle's ability to receive and execute commands. Critical commands usually require a manual override, so that the command is only executed in an appropriate condition. Just as two-step commands can be used to provide fault protection for hazardous commanding operations, such as ordnance firing, ground-based critical commands are used to reduce the risk of erroneously sending of potentially hazardous commands. Ground-based critical commands require additional authorization (manual permission to send command) before the command can be sent to the space vehicle. This additional authorization is used to help ensure that such commands are only executed in appropriate conditions.

4.1.3.4.7.2 Hazardous Constraint Violations

Onboard protection to prevent permanent damage resulting in either loss of mission or a degraded mission due to constraint violations should be provided. Such protection should be provided for sensitive payloads requiring need for protection from damaging environmental exposure.

Rationale: Environmentally-sensitive equipment that are protected by measures that are commandable (active compared to passive) can be damaged by an errant command to change the configuration (open compared to close).

Example: Refer to WIRE anomaly report, found in section 5.5.2.1 (Hardware Unit Testing).

4.1.3.4.7.3 Critical Constraint Violations

Onboard protection to prevent permanent damage resulting in either loss of mission or a degraded mission due to constraint violations may be provided. The ground may provide critical constraint violation protection.

Rationale: Some sensitive payloads require protection from the sun. Therefore, any ground-based or autonomous-maneuvering algorithms should avoid pointing the sensitive payload at the sun. Similar protection may be needed for any sensitive equipment that needs to be protected from any type of operating or environmental-exposure constraint.

4.1.3.4.7.4 Timers in Critical Applications

On board timers should be used carefully, understanding the possible consequences of a ill-timed event.

Rationale: Timers are useful for critical events such as onboard deployment sequences, fault management command sequences, and separation events. Incorrectly timed events (which are also difficult to test) can have catastrophic consequences.

Example: Quickbird Deploys Solar Panels Prematurely
Abstract
Initial reports were that the rocket's second stage shut down too early. But Russian officials maintain that Quickbird's internal computer was not reset after a one-hour launch delay. The solar panels were programmed to deploy a certain time after launch. After the delay, the panels deployed during launch leading to the failure.

Symptom
The payload deployed its solar panels during launch after its computers were not reset following a one-hour launch delay.

Cause
After a one-hour launch delay, someone forgot to reset Quickbird's internal computer. The solar panels were programmed to deploy a certain time after launch. They ended up deploying during the launch.

Lesson Learned
Remember to reset the spacecraft computer timer after a launch delay [6].

4.1.3.4.8 Design Utilization Guidelines

Excess design capability (design margin) can be used to provide protection for failures in non-redundant hardware such as solar arrays and batteries. Design margins can also be used to allow support to changes that may be needed after CDR or even after launch. The design utilizations listed here generally support fault management changes post-launch. Table 16 details the design utilization requirements applicability by space vehicle class.

4.1.3.4.8.1 Solar Array Power Margin

Fault management should provide input to the power subsystem on the amount of solar array power margin needed for the planned fault management modes.

Rationale: It is impractical to have fully-redundant solar arrays. However, solar-array fault protection should be provided by using strings of solar cells and sizing the array to allow a specified number of these strings to fail.

4.1.3.4.8.2 Battery Capacity Margin

Fault management should provide input to the power subsystem on the amount of battery charge capacity margin needed for the planned fault management modes.

Rationale: It is not always practical to have redundant batteries, thus the batteries should be sized to allow for a pre-determined number of cells to fail.

4.1.3.4.8.3 Processor Throughput Utilization

Fault management should provide input to the command and data handling and flight software subsystems on the amount of processor throughput utilization margin needed for potential fault management changes on-orbit.

Rationale: Processor throughput is normally stated to be no greater that 80% of its maximum capability at launch. This should be adjusted to allowing only a 50% margin to allow for future growth if the software definition is not mature or expected to be complex.

4.1.3.4.8.4 Processor Memory Utilization

Fault management should provide input to the command and data handling and flight software subsystems on the amount of processor throughput utilization margin needed for potential fault management changes on-orbit. Typical margins for processor memory are 50% at PDR, 40% at CDR, and 30% at launch.

Rationale: The memory used to store and execute the flight software. The flight software is the first line of defense when changes to the space vehicle are needed to adapt to an on orbit fault. The impact of a fault may require that additional functions are needed to return the space vehicle to an operational state. The memory is directly affected by changes to the flight software. Changes to the flight software usually result in added code to address the new space vehicle configuration. To allow flexibility in the execution of the required flight software changes, memory margins should be a managed commodity (and could be viewed as an expendable entity).

Examples: GPSIIR
Abstract: On 20 August 1998, Mahendra Wijesinghe (LMMS-VF) presented a briefing to 2SOPS, 1SOPS, The Aerospace Corporation, and NOSO to propose 11 SPU software patches based on lessons learned through the first year of on-orbit life for SVN043. On 3 December 1998, SMC/CZK formally turned LMMS on to work the Enhancement Patches. The CDU Fault Correction patch is designated SPU Software Version 1.1 Patch #33.

Prior to the Patch #33 load, REDMAN checks the health of the CDU by monitoring the CDU's power supply voltage. If the voltage is too low or high, the response is for REDMAN to command to the backup CDU. The CDU voltage telemetry is sent to the SPU via the TIU. A failure mode has been determined by which a partial failure of the TIU could cause the CDU to look like it failed. In this case, switching CDU's would not correct the problem. Adding an enhancement to switch the TIU first, before switching CDUs would correct for this potential failure.

Symptom: In November 1998, a single point failure (SPF) condition was identified on SVN055. The IIR design powers both KIRs through the Selected CDU power supply. If this power supply fails, the vehicle will not accept any ground commands. All IIR vehicles, except SVN043, have had a harness change such that both KIRs are powered by the bus directly and are not dependent on power from the selected CDU.

On 4 December 1999, Patch #40 was uploaded to SVN043 to avoid the SPF possibility on SVN043.

With only Patch #40 loaded, if a TIU were to fail, the vehicle could be left with carrier and no telemetry after a CDU and SPU swap.

Cause: Subsystem(s) Involved: SPU Flight SW.

In summary, a REDMAN test failure derived from telemetry could be the result of a single telemetry channel failure in the TIU. Therefore, the fault-protection software should switch TIUs as part of the recovery algorithm. This logic applies to all REDMAN tests that use telemetry as the basis for the health and status check.

Corrective Action: As part of the follow-on corrective action for this condition, Patches #28, 30, 33 and 40, and Macro #24 and 25 development was expedited. The patches and macros were demonstrated on the IIR Telecomm Simulator on 21 January, and successfully loaded on SVN043 on 5 March 1999.

After the Emergency load of Patch #40 to SVN043, the Follow-On Course of Action involved expediting development and delivery of SPU Patches #28, 30, and 33. With Patch #40 loaded, there was a potential that a TIU failure could cause loss of telemetry or loss of frame sync to the SPU (the same symptoms as a CDU power supply failure) and leave SVN-43 in a recoverable, but awkward configuration. Patch #30 maintains the SPU to CDU command capability after REDMAN has exhausted all means of switching components (including Control SPU) to correct test failures. Patch #33 inserts a TIU swap (in addition to the CDU swap) as a corrective measure if a major or minor frame fails to transition, if the CDU power supply voltage exceeds expected limits, or if High Charge Rate commands have failed to execute.

As of 5 March 1999, all of the proposed corrective actions to avoid potential problems with SPU Subcomm data, and the CDU-KIR power SPF have been addressed. The loads include: SPU Patches #28, 30, 33, and 40, Macros #24 and 25, and some Rdmgt SAD GSP uploads to SVN043 [6].

4.1.3.4.8.5 Databus Utilization

Fault management should provide input to the command and data handling and flight software subsystems on the amount of databus utilization margin needed for potential fault management changes on-orbit. Databus utilization should be specified to allow for future growth or modifications.

Rationale: The databus utilization should be no more than 70% at launch. The databus used to transfer sensor data or actuator commands is affected directly by on-orbit software changes to adapt to the new space vehicle following a vehicle fault. The result of software changes to adapt to on-orbit failures is usually an increase or re-sequencing of the databus traffic. Databus traffic should not be viewed as an average over the processing cycle, but more appropriately a quantized entity with limitations and constraints on the time and size of transfers and their associated conflicts. Margin in the right places in the processing cycle is a more useful concept to consider when designing in margin. When the margin is harvested following a fault, usually additional telemetry is needed and/or additional commands are needed. Telemetry can generally, except for the most stringent of requirements, be sampled anytime during the processing cycle. Commanding on the other hand tends to be quantized based on the processing in the software and command grouping, i.e. it is not asynchronous.

Example: The space based infared satellite system (SBIRS) experienced a test failure when the fully redundant command and data handling subsystem was integrated and tested with flight software. The test failure identified a flaw in the architecture where a databus used between the command and telemetry unit (CTU) and the attitude control unit (ACU) could not support the data throughput required. Although the

hardware was capable of meeting the data throughput requirement, the software management of the bus resulted in large transfers that could not be completed by the next processing epoch. The interface timing analysis did not account for how the software intended to use the databus [10]. Significant flight software, fault management, and testing impacts occurred, delaying the program.

4.1.3.4.8.6 Expendables Margin

Fault management should provide input to the propulsion subsystem for propellant and catalyst beds should be sizing to accommodate an amount of usage reasonably expected for safe/survival mode entries. *Rationale:* Unbudgeted use of expendables can shorten the mission time of a space vehicle. Safehold mode designs that use thrusters needs to account for the use of expendables when budgeting propellant load and for unanticipated problems.

Example:
Abstract
Japan's Nozomi Mars probe made a lunar flyby on 18 December 1998 followed by an Earth flyby on 20 December. The perigee burn went wrong and apparently resulted in an orbit with a period of about a day; reportedly a second flyby on 21 December correctly placed it in solar orbit. Engineers worried, though, that they may have used too much fuel when correcting the probe's orbit to continue the mission planned. Professor Ichiro Nakatani of the government-run Institute of Space and Astronautical Science stated that more fuel was consumed than expected and a study of the effects from this problem will be done and there is a possibility that a change in the original plan will be forthcoming. The $80 million Planet-B probe, renamed 'Nozomi' or 'Hope' after liftoff was scheduled to reach Mars late in 1999.

Symptom
The perigee burn went wrong and apparently resulted in an orbit with a period of about a day; a second flyby on 21 December correctly placed it in solar orbit. Engineers worried, though, that they may have used too much fuel when correcting the probe's orbit to continue the mission planned.

Cause
During the escape burn the anti-back-flow valve did not open fully, causing the engine not to deliver the required thrust. Two further burns the next day improved the trajectory, but in the process left the spacecraft with insufficient propellant to enter orbit around Mars

Recovery
The mission analysis team devised a recovery strategy. By flying past Mars in 1999, and remaining in heliocentric orbit, the spacecraft would be able to exploit Earth flybys in December 2002 and June 2003 to encounter Mars again in December 2003 at a slower relative velocity than originally planned, which it would be available to overcome with the propellant available. The penalty, however, would be spending an additional four years in space [6].

4.1.3.4.9 Autonomous Fault Protection Guidelines

This section describes the detailed autonomous fault protection requirement guidelines that drive the fault management architecture. Table 17 details the autonomous fault protection guidelines applicability by space vehicle class.

4.1.3.4.9.1 Ground Command Response Initiation

Fault responses should be capable of being initiated by ground operator control.

Rationale: Ground should be allowed to command the space vehicle into any contingency mode, the general purpose safe mode or the low-power response mode to put the space vehicle into a safe state.

4.1.3.4.9.2 Fault Response Enable/Disable Capability

Where applicable, fault responses should have two enable/disable mechanisms (or the functional equivalent):

1. An enable/disable of the stored command sequence, if stored command sequences are used, and
2. An enable/disable by ground control or by fault protection/deployment algorithms.

All monitors should include an enable/disable mechanism or the functional equivalent.
Each enable/disable should be specified by a single parameter unique to each fault protection algorithm.

For critical events, enable/disable strategies may be used to minimize or prevent the effects of an erroneous fault indication [5], e.g., planetary insertion burns.

Rationale: Fault response enable and disable capability is needed to configure fault protection for the vehicle modes during activation or during specific mission operations. Also, the fault protection configuration may need to be modified during mission critical events, such as deployments, or to prevent spurious activations.

4.1.3.4.9.3 Fault Monitor Enable/Disable Capability

All fault monitors should include an enable/disable mechanism or the functional equivalent [7].

Rationale: Some fault protection architectures have multiple fault monitors feed into a single-fault response. In this case, it is necessary to be able to individually enable or disable each fault monitor without disabling the fault response to modify fault protection during mission critical events, such as deployments or thruster maneuvers, or to minimize or prevent the effects of an erroneous fault indication.

4.1.3.4.9.4 Fault Detection

The fault management system must first detect a fault to be able to determine what response is appropriate. Hardware and software detection sources should have two criteria [5]:

- *Direct detection.* Detection mechanisms should be as direct as possible (i.e., a direct measurement is preferred over a calculated or derived measurement).
- *Detection coverage.* Detection mechanisms should only be required to detect a failure to the level at which that failure can be isolated or corrected.

Rationale: The fault protection system must first detect a fault to be able to determine what response is appropriate. Hardware and software should provide the measurands needed to detect the faults.

4.1.3.4.9.5 Response Initiation

Fault responses should be initiated if and only if space vehicle performance is unacceptable, or there is a significant risk to the mission or to subsystem safety [5]. Fault responses should be capable of being initialed under ground control.

Rationale: Ground should be allowed to command the space vehicle into a contingency mode, safe mode or the low power response to safe the vehicle.

4.1.3.4.9.6 Fault Protection Modifications

All fault protection parameters which may reasonably be expected to change as a function of mission mode, type of activity, fault history, or operational experience should be alterable by ground control without requiring flight software modification [5]. Fault monitor and responses should be modifiable by ground. Safe-hold mode entry and exit criteria should be modifiable by ground. Any response delay timers should be modifiable by ground.

Rationale: Rationale: Fault protection parameters that may require frequent or known changes during the mission should be easily modified by ground command. Fault monitor and response parameters include, thresholds, persistence counters, delay timers, etc. Fault identification methods currently available are not infallible, and new faults may need to be detected and responded to. Also, operational experience may uncover a defect in a fault monitor or response, and a new algorithm may need to be provided. Significant fault protection modifications or unexpected parameter modifications can be modified by uploading flight software code or data parameters.

4.1.3.4.9.7 False Alarm/Spurious Signal Negation

Software-implemented fault-detection monitors and subsystem internal-fault protection should provide spurious-signal negation (false-alarm mitigation).

Software monitors used by system and subsystem internal-fault protection should have the following features to mitigate false alarms:

- *Monitor thresholds*. Where possible, thresholds should use reasonableness checks, detection filtering (to exclude certain faults from a previously established fault database), or redundant detections [7]. Thresholds should be biased toward false positives (unnecessary safing) compared to false negatives (not responding to a hazardous condition). Thresholds can always be adjusted on orbit to minimize recurrence of a false positive, whereas a false negative can end the mission.
- *Threshold modifications.* Monitor threshold values should be alterable by ground or sequence command, or by fault protection responses as appropriate. As a goal, monitors are best designed to detect and disregard failed sensors [7].
- *Redundant detection.* For detections where an inadvertent trip would result in a severe response (for example, downlink loss, irreversible hardware swaps, large use of expendables, critical sequence cancellation), and where a sensor anomaly could cause an inadvertent trip, independent physically or functionally redundant detections are employed such that simultaneous detections are necessary for response initiation [7].
- *Fault response tolerance.* Monitors are designed to be tolerant of off-nominal conditions following a reconfiguration resulting from a fault protection response [6] (e.g., thresholds might be relaxed as part of a response).
- *Persistence Counters.* Monitors should use persistence counters to prevent spurious detection and fault responses.

Rationale: It is often necessary for fault responses to reconfigure hardware, that is, to power off suspect equipment and power on unused equipment, to contain a fault. Other fault responses suspend mission

operations by entering either a contingency or safe mode and may power off equipment to 'protect' the vehicle. To reduce the risk of a failure occurring upon power off or power on, healthy equipment should not be turned off unless the space vehicle is at risk. Therefore, fault detection monitors need to be designed to minimize the occurrence of unnecessary hardware reconfiguration and interruptions to mission operations due to false positives while at the same time protecting the hardware from fault negatives. Spurious signal negation techniques are used to reduce the occurrence of false positives causing unnecessary fault response activations.

4.1.3.4.9.8 Fault Response Timeliness

Fault detection monitors, thresholds, delay timers, and persistence counters should be set to detect and respond to the anomalous condition before the effects can progress to the point of causing unrecoverable damage.

Rationale: For fault management to perform correctly, it is necessary for the monitors, thresholds, any delay timers, and persistence counters to be set appropriately so that the fault is contained before any damage to hardware can occur. The time to critical effect (TTCE) is the minimum timeline duration in which an anomaly can manifest its effect if not contained. The time to irreversibility (TTI) is the minimum timeline duration for which an anomaly should be responded to. When setting thresholds, delay timers, or persistence counters care should be taken so that TTI is always a shorter duration than the time to critical effect.

4.1.3.4.9.9 Fault Responses False Alarm Tolerance

Unintended entry into a fault protection response in the absence of a fault must not present a hazard to the space vehicle or mission [7].

Rationale: Even if precautions have been take to minimize fault response execution due to false alarms, such an event cannot be eliminated entirely. Therefore, an unintended execution of a fault response should not pose a hazard to the space vehicle.

4.1.3.4.9.10 Return to Previously Swapped Equipment

Autonomous hardware failover responses that allow a return to previously swapped equipment should be avoided. However, if deemed necessary to protect against identified faults, the failover logic should have a default state of disabled and the ability for ground to command it to disabled.

Rationale: On-orbit radiation effects experiences have shown that some radiation induced faults can be cleared by hard-power cycling the equipment. This is called single-event latch-up. Generally, Class A and B space vehicles are not allowed to use devices that can latch-up. To provide protection against latch-up, some failover logic provides the ability to swap back to the primary processor. The default state should be enabled to prevent processor power cycling following dead battery recovery. Also, the ground must have the ability to turn this failover response off. If such a response is used, failure modes that would cause the response to activate must be considered and the response should not pose a risk to the mission.

4.1.3.4.10 Diagnostic Data Guidelines

The space vehicle design should ensure that following an anomaly, space vehicle telemetry will be sufficient to perform the preliminary fault isolation and analysis required for ground control to perform near-term corrective or recovery actions [5]. Telemetry data is necessary to be able to determine the space vehicle health and perform fault isolation. Telemetry requirements necessary to diagnose anomalous

behavior and to provide a history of autonomous actions that have occurred are provided in the following paragraphs. Table 7–10 details the diagnostic data guidelines applicability by space vehicle class.

4.1.3.4.10.1 Telemetry Coverage

Telemetry coverage should be provided to the extent possible. If at all possible, telemetry coverage should be provided during mission critical events.

Rationale: Telemetry data is necessary to be able to reconstruct events leading up to an anomaly or failure. When possible, 4 π steradian telemetry coverage should be provided.

Examples:
Abstract
On 15 August 2002, communications with CONTOUR was lost after the solid rocket motor (SRM) burn, which was supposed to place CONTOUR into the comet encounter trajectory. CONTOUR most likely broke apart due to excessive plume heating during the SRM burn. Improper thermal analysis was blamed

Symptom
CONTOUR ceased downlink after the scheduled telemetry blackout, near the period of the solid-rocket motor burn. Three objects were spotted, indicating that the vehicle broke up.

Cause
The NASA mishap investigation board concluded that the most likely root cause was structural failure of the spacecraft during the solid-rocket motor burn. The SRM nozzle was deeply embedded within the spacecraft, but the designers failed to adequately account for plume heating. The plume analysis was based on a model constructed by Orbital Sciences and reported in a paper, but the paper had an error. Had drawings been compared to the original model or qualification testing been performed, the flawed design would have been found.

Recovery
Repeated contact attempts were made to no avail.

Corrective Action
The Board made the following recommendations: (1) Have telemetry coverage during all ‘Critical Events’ (2) Consider day-to-day oversight of Project Investigator (PI) led missions by a field center; (3) Ensure that the Prime Contractor has adequate sub-contractor oversight and verification; (4) Develop a solid-rocket motor-thermal design guide; (5) Identify expertise limits and augment with proper personnel; (6) In areas of limited expertise, devote stringent attention to requirements; and (7) Have subcontractors ‘Buy In’ to how their products will function within the overall design.

Lessons Learned
(1) Double check all analysis models, assumptions, methods and predictions; (2) Develop a rigorous process for using experience as a basis for accepting further designs and equipment; (3) Have the original analyst review the final product; and, (4) Make sure the key subcontractors accept how their product is being used [6].

4.1.3.4.10.2 Real-Time Telemetry

At a minimum, the real-time telemetry stream should include the following:

1. Measurements to unambiguously identify the current engineering subsystem configuration and operating status,
2. Enable/disable states of fault protection software, and enable/disable states of fault protection dedicated hardware,
3. Active/inactive states for fault protection software responses,
4. Identification that a fault response has been activated,
5. Monitor detection logic output (regardless of whether the monitor is enabled or disabled), and
6. As appropriate, cumulative counters of faults detected.

Telemetry design should ensure reasonable and timely sampling frequencies for these items [7]. Following a fault, space vehicle health and safety telemetry, not payload telemetry, should be the priority. This includes not just the items to be sampled, but the frequency at which they are sampled [5].

4.1.3.4.10.3 Stored Telemetry (TLM) for Out Of View (OOV) Operations

Space vehicle state of health telemetry data should be stored and made available for later download when direct downlink telemetry coverage is available.

Rationale: Storing space vehicle telemetry on board when downlink coverage is not available allows the later (when downlink coverage is available) downloading of the space vehicle telemetry for examination of space vehicle health during the out-of-view period.

Example:
Abstract
Experience with SVN43 during anomalous situations has shown that an on-board capability for buffering data in spare memory, over periods when the space vehicle isn't in contact with the ground, would greatly increase the engineering communities' ability to analyze these situations. Additionally, this ability to record data could be coupled with a 'stop' switch that could be set by the ground to stop the data collection after an event has occurred. This kind of an enhancement would speed up anomaly resolution and add to the engineering staff's ability to perform anomaly resolution. The data would be stored in the SPU's unused memory, so there would be minimal effect on vehicle resources. The data could be retrieved via a memory dump for post-past analysis and would minimize the amount of personnel and MCS resources being utilized during anomalous situations

Symptom
During anomalous situations has shown that an on-board capability for buffering data in spare memory, over periods when the space vehicle isn't in contact with the ground, in spare memory, would greatly increase the engineering communities' ability to analyze these situations. Additionally, this ability to record data could be coupled with a 'stop' switch that could be set by the ground to stop the data collection after an event has occurred [3].

4.1.3.4.10.4 Anomaly Telemetry Data Storage

Engineering data prior to, during, and after the execution of any fault response should be stored on-board and made available for later downlink. Some exceptions are made for recorder and command subsystems failures. Data variations close to (faster) the time of response should be stored at as high a rate as possible. Conversely, data variations far (slower) from time of response can be stored at a lower rate.

Rationale: For many systems, solid-state data recorders may not be available early in the activation phase, or may need to be powered off to save power during a low-voltage condition. Following an

anomaly involving a fault protection activity, the space vehicle design should ensure that sufficient information is available to reconstruct the audit trail— the sequence of fault protection events following the anomaly. On-board data storage should capture, at a minimum, the data needed to reconstruct the sequence of fault response events. Anomaly data is usually stored at a high as possible rate and is limited to storing data for a short time before and/or after the fault response activation.

4.1.3.4.10.5 Fault Management Mode Telemetry

The space vehicle design should ensure that following an anomaly, space vehicle telemetry will be sufficient to perform the preliminary fault isolation and analysis required for ground control to perform near-term corrective actions [7].

Rationale: Upon entry to any of the fault management modes (contingency or safe mode), the real time telemetry stream still should contain the similar data as the normal mode telemetry. However, following a fault, space vehicle health and safety telemetry, not mission data should be the priority.

4.1.3.4.10.6 Fault Response Diagnostic Data

Following an anomaly involving a fault protection activity, the space vehicle design should ensure that sufficient information is available to reconstruct the audit trail—the sequence of fault protection events.

Rationale: Diagnostic data is needed to analyze the anomaly and to identify the sequential effects of the anomaly on space vehicle performance. At a minimum, diagnostic data should include component or unit state information, passive fault response status telemetry, status of hardware used in voting schemes, unique identification and time-tagging of monitors and responses which have been activated. It is also desirable that diagnostic data include the value of the measurement which initiated a response [5].

4.1.3.4.10.7 Preservation of Stored and Diagnostic Data

Fault responses should ensure that diagnostic data is not overwritten as the result of a fault response and that any critical data stored onboard is not destroyed.

Rationale: Circular telemetry buffers can be used to store telemetry at a high rate or used to store fault management detection and response data. When circular telemetry buffers are used to capture data prior to an anomaly, it is important to save the oldest data following the response, to ensure data immediately following the response isn't overwritten.

4.1.3.4.10.8 Hardwired Critical Telemetry

The space vehicle should be capable of providing critical telemetry sufficient to determine space vehicle operating state independent of flight processor functionality.

Rationale: Some programs have telemetry formatting implemented in space vehicle. Although software formatted telemetry allows for easy modification, it also couples telemetry availability with attitude control activities. Critical operational status telemetry and telemetry sufficient to diagnose the processor should be available in the event the flight computer is not operational. Telemetry points to consider are critical temperatures, battery status, command counts, and processor status.

4.1.3.4.10.9 Processor Memory Dump Capability

Space vehicle architectures and processor designs should allow the ground to download processor memory.

Rationale: The onboard processor usually contains the fault management software. The onboard processor has the best visibility on the fault that was detected (limit that was exceeded) and the chosen response. The ability to download (dump) processor memory provides the opportunity to determine if there are nuances to the detection of the fault or the response that may need to be changed to better protect the space vehicle as there may be limitations to the amount of information available in telemetry saved or sent to the ground (link rate).

4.1.3.4.10.10 Archive Telemetry

Telemetry received by the ground station, should be archived for later use in anomaly resolution.

4.1.3.4.10.11 Archive Command History

Command history should be archived for later use in anomaly resolution.

4.1.3.4.11 Damage Minimization Features

Damage minimization features can be used to provide protection for various engineering design common-cause faults. Effective damage minimization features are provided below. These features have been used many times to overcome anomalies and either allow the satellite to be fully recovered and put into service or to allow recovery of some mission aspects. Table 19 details the damage minimization features requirements applicability by space vehicle class.

4.1.3.4.11.1 Maximize Propellant Load

Final propellant load should utilize full tank capability (fill up the gas tank before launch).
Rationale: Many space vehicle have been able to use propellant at the expense of service life for at least partial recovery of the mission following booster under-performance.

Example:
TDRS 1 was launched on 4 April 1983 as STS-6. An inertial, upper stage (IUS) was used inject TDRS 1 into its operational orbit. However, an IUS failure during the second stage burn caused the IUS/satellite stack to tumble end-over-end. Ground controllers were able to release the satellite, which stabilized itself. The satellite's propulsion system was then used to place the satellite into its operational orbit [3].

Example:
The Hiten satellite, after underperformance by the launcher, used onboard propellant to restore its orbit after poor performance from the booster. By robust design, the Hiten vehicle included the additional propulsion capability in case such an event occurred.

Other satellites have used similar strategies to make up for being placed in a lower than expected orbit. Additional examples are Hipparcos, GOES 1, and Artemis [3].

4.1.3.4.11.2 Re-programmable Processor

The space vehicle design should allow the processor to be re-programmed by ground operators.

Rationale: To react to on-orbit conditions and anomalies, uploading patches to the flight software is the most feasible method for accommodating changes in the space vehicle fault management.

Example:
Abstract
Experience with SVN43 during anomalous situations has shown that an on-board capability for buffering data in spare memory, over periods when the space vehicle isn't in contact with the ground, in spare memory, would greatly increase the engineering communities' ability to analyze these situations. Additionally, this ability to record data could be coupled with a 'stop' switch that could be set by the ground to stop the data collection after an event has occurred. This kind of an enhancement would speed up anomaly resolution and add to the engineering staff's ability to perform anomaly resolution. The data would be stored in the SPU's unused memory, so there would be minimal impact on vehicle resources. The data could be retrieved by a memory dump for post-past analysis and would minimize the amount of personnel and MCS resources being utilized during anomalous situations.

Symptom
During anomalous situations has shown that an on-board capability for buffering data in spare memory, over periods when the space vehicle isn't in contact with the ground, in spare memory, would greatly increase the engineering communities' ability to analyze these situations. Additionally, this ability to record data could be coupled with a 'stop' switch that could be set by the ground to stop the data collection after an event has occurred.

Cause
Subsystem(s) Involved: SPU Flight S/W

Recovery
SPU patch #37 was loaded to SVN043 on 10 September and SVN046 on 7 October 1999.
The SPU enhancement proposal including the rolling buffer (Patch #37) was presented on 20 August 1998. The Patch Tapes were delivered to 1 SOPS and 2 SOPS on 26 May 1999 [6].

Example:
GPSIIR SVN043 needed 11 software patches uploaded based on lessons learned through the first year of on-orbit life. These software patches corrected for oversights in fault management [6].

4.1.3.4.11.3 Polarity Error Correction Capability

Space vehicle designs should provide the means to either correct or compensate for incorrect control-loop or link-tracking loop polarity.

Rationale: Sign errors are a frequent cause of attitude control anomalies. There are numerous examples of polarity errors causing attitude control anomalies despite design reviews and ground testing designed to detect polarity errors. Thus, it is necessary to provide the capability into the design to allow ground to reconfigure the control loop to compensate for polarity errors. Polarity error compensation should not be limited to only attitude control loops. Antenna tracking control loops and valve wiring are also subject to polarity errors and polarity testing escapes due the test limitations or human error.

Example: The total ozone mapping spectrometer in NASA's earth probe series (TOMS-EP) was a small satellite and was launched on 2 July 1996 and inserted into a sun-synchronous orbit. Concern about the polarity of the attitude sensors was raised at the critical design review, and a polarity test plan was developed, reviewed, and approved. The test plan called for checking polarity at the component level, and a system-level polarity check would methodically test the sensors with known inputs and verify the

responses of the actuators. Shortly after launch, it was found that two of the sun sensors had been cross-wired. This problem was solved by switching the sensors in software. The error was not detected during testing because the sensors were not installed in their flight configuration on the solar arrays. Later, it was found that the polarities of the magnetic torquers were inverted. This polarity escape was attributed to an error of interpretation. During testing, two axes had been determined to be incorrect, and so were reversed into, what ultimately proved to be, an incorrect configuration. This polarity error was also resolved by a software update and the TOMS-EP was entered into service [6].

Example:
Abstract
The TIMED (Thermosphere, Ionosphere, Mesosphere, Energetics and Dynamics) spacecraft experienced several anomalies after deployment, the most important being tumbling and an inability to point the spacecraft to the sun (see ARN x02102802). Tumbling occurred because torquer algorithms were 180 degrees out of phase in the flight software. A sequence of maneuvers, turn-offs, and software uploads fixed the problem. If proper management and test processes had been in place, the error would have been found prior to launch.

Symptom
TIMED experienced four anomalies immediately after separation: (1) failure of the spacecraft to stabilize; (2) failure of the spacecraft to acquire sun-pointing safe-mode; (see ARN x02102802); (3) failure of the spacecraft to achieve 0.5 degree pointing requirements; (see ARN xx02102803); and (4) star tracker AST 1 tripping off (see ARN x02102804). In addition, two of the four instruments, SEE (Solar EUV Experiment) and TIDI (TIMED Doppler Interferometer), experienced anomalies on orbit. Recovery of the spacecraft was achieved within three months after launch

Cause
The Investigation Board found several project management deficiencies. The torque phasing was set, not by analysis but by test. During the test the ACS engineer was confused with the earth's magnetic pole configuration (the Earth's magnetic south pole is actually in the Arctic, not in Antarctica). As a result the polarity of the magnetic torquer bar currents was reversed for all three axes in the Guidance and Command software. Instead of damping the spacecraft motion, the software caused it to increase. Test procedures with previously defined limits or a peer review of the proposed software changes could have found the problem.

Recovery
A software patch was uploaded to correct the polarity. Once the sign fix was implemented, the momentum began decreasing. Stability was achieved about 180 minutes after separation.
Corrective action: Program Management, Systems Engineering, and Product Assurance should provide sufficient protective processes, i.e., formal configuration control, peer reviews, and failure reporting at an earlier stage of the development process.

Lesson learned
If proper management and test processes with limits had been in place, the error would have been found prior to launch [6].

4.1.3.4.11.4 New Technology Safe Guards

Backup systems may be used to minimize the risks of the new technology insertions.

Rationale: The newer the technology, the higher the risk of unintended effects, degraded performance, or failure. Fault management can compensate for new technology insertion risk by employing qualified,

existing equipment as a backup (with perhaps less than desired performance) to ensure some capability is maintained.

4.1.3.5 Fault Management Verification Guidelines

Fault management verification planning should be performed early in the program as part of the overall program verification planning activities. The fault management requirements should be verified by appropriate analysis and test. In performing fault management verification planning, it is important to identify at what level the requirements are to be verified at and to identify what the associated test environment requirements are. Specific fault management verification guidelines are provided in the following paragraphs. The various fault management verification approaches, test environments, and analyses are described in section 5.5 (fault management Verification Techniques), along with rationale for when determining the level of fault management verification activities and determining what analysis should be performed.

Table 20 details the fault management verification requirements applicability by space vehicle class.

4.1.3.5.1 Fault Management Performance Verification (Test Like You Fly)

The fault management verification effort should verify the performance of the fault management actions when exposed to identified faults or failure symptoms [11]. Test-like-you-fly (TLYF) concepts should be applied to fault management performance testing regardless of test level. All fault management performance testing should be performed in flight-like configurations to help verify fault management design and identify incorrectly set thresholds. TLYF also requires the test environment to provide as flight-like an environment as possible. In many cases sticking to true TLYF goals in verifying fault management design requires conducting failure mode and effects testing (FMET).

Regardless of fault management implementation methods used or level of complexity, FMET concepts should be used for all fault management testing.

Rationale: Traditional verification techniques focus on showing that the implementation satisfies the specification and that the specification satisfies the higher-level requirements, as shown in Figure 8. Although it may seem that this same approach can be applied to fault management, this process is not adequate. The fault management implementation may result in unintended functions which may be hazardous to the space vehicle. These potential unintended functions will not be caught by the traditional one-way verification process, where the implementation is shown to satisfy the functional specification [12]. Because fault management is intended to enhance space vehicle safety, it is necessary to show that the implemented fault protection approaches will result in the expected fault containment. In other words, verify that the fault management implementation provides the expected fault protection for the identified faults, as shown in Figure 9. Such an approach will at least show expected performance for the fault protection for the identified faults [12].

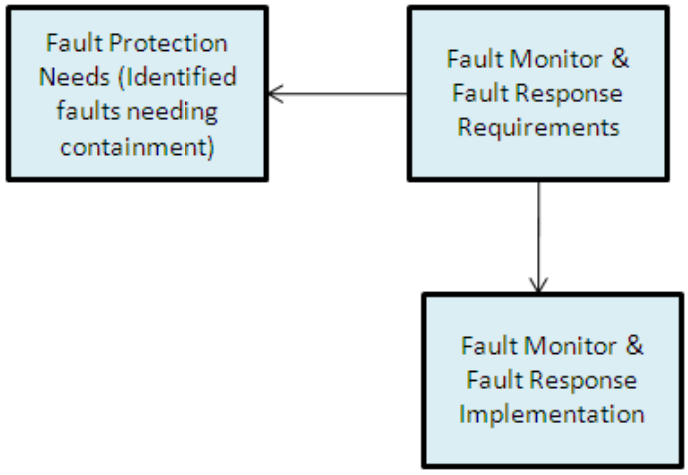


Figure 8. Traditional verification approach.

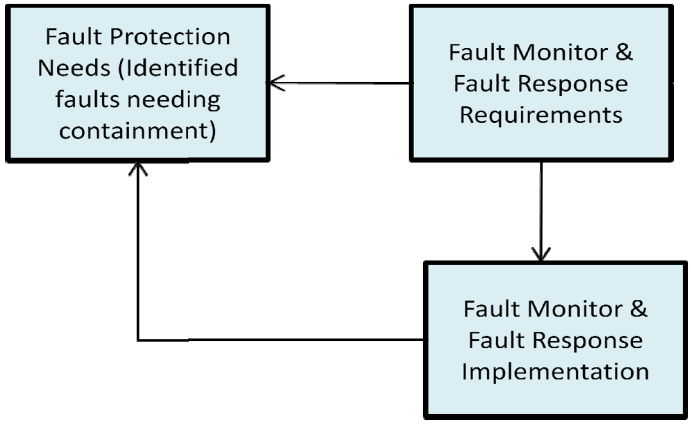


Figure 9. Recommended fault management verification approach.

4.1.3.5.2 Fault Management Logic Paths

All fault management logic paths should be verified by test or demonstration.

4.1.3.5.3 Verification by Similarity

Verification by similarity should never be used for fault protection monitors, thresholds, persistence, or responses.

Rationale: Once a fault protection monitor, response, or mode is included in a space design, even for the same use, it should be re-analyzed and/or re-tested and re-verified to ensure subtle differences in design do not reduce the effectiveness of the function it is to perform.

Examples where verification by similarity have caused catastrophic failures include Lewis 1997 where use of a heritage ACS design from TOMS-EP resulted in an anomalous condition due to an undetectable axis using only two gyroscopes. The ACS design responded in a manner which it would have used the third gyroscopes [3]. The failure review board concluded, “The ACS verification process failed to address the deficiencies that might arise from an improper application of software designed for a much different

space vehicle." Lesson: When heritage hardware and software is used, make sure its use is well warranted and use cases verified [13].

4.1.3.5.4 Worst Case Circuit Analysis (WCCA)

All hardware-implemented fault tests and response verification should include worst-case circuit analyses to verify that over life, the expected operating environment (temperature and radiation exposure), part tolerance variations, and the circuit performs correctly and has the proper threshold settings to ensure that the fault is detected and not produce false alarms.

4.1.3.5.5 Threshold, Persistence, Interference (TPI) Analysis

A threshold, persistence, and interference analysis should be performed as part of the fault-protection verification activities. The TPI analysis should verify that the TTI is less than the TTCE.

Rationale: It is necessary to verify that the fault protection monitors have the correctly-set thresholds and persistence counters to ensure that the faults are detected and responded to in a timely manner to contain the fault so that irreversible damage to the space vehicle does not occur. It is also necessary to verify that fault responses do not adversely interfere with each other.

4.1.3.5.6 Contingency Mode Analyses

Each implemented actively controlled contingency mode should have analysis showing that power consumption, battery charge and thermal stability are provided. Contingency-mode attitude control analyses should show that the desired contingency-mode attitude can be captured for the fault conditions, as well as showing that performance requirements are met. Entry into these modes as a result of a failure causing the mode entry should also be verified by ACS simulations and closed-loop test under worst-case dynamic conditions.

4.1.3.5.7 Safe Mode Analyses

Safe mode should have analysis showing that power consumption, battery charge and thermal stability are provided. Safe mode attitude-control analyses should show that the desired safe mode attitude can be captured for the fault conditions, as well as showing that performance requirements are met. Entry into Safe mode as a result of a failure causing the mode entry should also be verified by ACS simulations and closed-loop test under worst-case dynamic conditions.

4.1.3.5.8 Low Power Response Analysis

The low power response should have analysis showing that the space vehicle can survive and be recovered from the response, provided that the initial fault did not result in both primary and redundant equipment failures.

Rationale: In extreme cases the space vehicle should have an low-power response that sheds the use of all non-essential power. For some space vehicle, this can include attitude control. This technique aids in allowing additional time for ground anomaly response by allowing longer life for sensitive components requiring environmental control, provided these components are protected from adverse environment exposure.

Fault management analysis efforts should ensure that cross-subsystem interactions along with environment effects are taken into account during severe anomalous conditions. For instance if a space

vehicle loses attitude control or experiences a bus short leading too little or no battery power, final settling states need to be determined for sufficiency to ensure the space vehicle reaches a safe and recoverable state. Attitude settling times and orientations along with worst-case power and thermal analysis can shed light on conditions where changes can be made to properly adjust space vehicle fault management responses, thresholds, and ground contingency responses. Bottom line: If the space vehicle can survive the severe anomalies that place the space vehicle in a worst case state, less severe anomalies should be recoverable.

4.1.3.5.8.1 Worst Case Power & Thermal Survival Analysis

The low-power-response analysis should include worst case power and thermal survival analyses.

Rationale: Determining potential space vehicle hazards due to the coupling effects of thermal and attitude instabilities and how they relate to the ability of the space vehicle to generate, store and deliver power to the space vehicle subsystems during severe anomalies is critical to evaluate. Some questions the analysis should answer are as follows:

- How long will the batteries provide sufficient power for thermal stability and basic communications while in modes where ACS is not available for worst-case orientations and roll rates?
- For power consumption levels in safe mode given worst-case roll rates (where there is minimum solar illumination on the array for power generation), is the power balance positive for worst-case scenarios?
- When space vehicle enters a response with no attitude control, what is the worst-case duration before components surpass their thermal-design limits?
- Do the critical contingency procedures reflect realistic reaction times in order to safe the space vehicle given the orbit (given worst-case communication opportunities)?

These analyses need to be revisited after any future updates to the thermal predictions, usually after thermal vacuum testing, or changes in the space vehicle mass properties. Slight adjustments in thermal predictions when extrapolating worst-case conditions can create short response times to deal with critical components.

Additional analysis can be conducted for potential failures for the most thermally sensitive components.

4.1.3.5.8.2 Steady State Analysis

The low-power response analysis should include uncontrolled, steady-state attitude analysis.

Rationale: In lower earth orbits the gravity gradient works to orient the space vehicle's maximum moment of inertia axis along the orbit's radial vector, much like the long axis of a dumbbell aligned to point at nadir. Depending on the solar array configuration for fixed arrays or the last orientation for tracking arrays, an analysis for the space vehicle settling times and final orientation possibilities will aid in understanding how to design and handle severe anomalies. For low-earth orbiting space vehicle, a steady-state analysis should be performed to show the orientation of the space vehicle when in a steady (uncontrolled) state.

Some questions the analysis should answer are as follows:

- How long does it take for the space vehicle to settle given worst case roll rates?
- Can the space vehicle settle in an orientation where the solar arrays are pointed away from the sun?
- Can the space vehicle settle in more than one orientation?

In high-altitude orbits and for space vehicles with large surface areas, other factors may contribute much more than earth's gravity to the settling times and orientations while in an uncontrolled state. Solar pressures in geosynchronous orbits play a large role in space vehicle orientations while uncontrolled and need to be carefully considered.

4.1.3.5.8.3 Uplink and Downlink Visibility Analysis

An uplink and downlink visibility analysis should be performed for all of the implemented fault management contingency modes, safe mode, and the low-power response.

4.1.3.6 Fault Management Sustainment

As stated earlier the fault management design and development process is an iterative and imperfect process. As a result, new faults may be identified throughout the program life cycle, including mission operations. The fault management documentation, including fault trees, algorithm description documents, fault branch verification matrices, databases, and supporting analyses should be updated whenever the design is changed. In addition, any modified database values, fault monitors, or responses should be re-verified with the same rigor as the original design.

The following paragraphs describe key fault management support that should be available after and continue after CDR and throughout the life of the space vehicle.

4.1.3.6.1 Maintain Fault Tree and Fault Management Verification Products

The fault tree or equivalent method of tracking identified faults along with any associated fault management verification data (including monitors and responses) should be maintained and updated as needed.

4.1.3.6.2 Maintain Closed Loop Test Bed

The closed loop test bed should be maintained and kept available after launch to support either updates in the fault management design or anomaly resolution.

Examples as to why this capability is needed are provided below.

Example:
An interplanetary probe recovered from a major anomaly. The spacecraft, designed to rendezvous with an asteroid, employed extensive autonomy because ground intervention during an emergency would take too long. The designers studied the history of an earlier project, which terminated prematurely after a data error depleted on-board fuel.

Three years into the flight, an engine burn aborted. A missing command in the burn-abort contingency command script prevented a graceful transition into the safe mode, and a series of anomalies ensued. Communication was lost for 27 hours before the flight computer regained control.

The initial script error was not caught during software tests. Hardware-in-the-loop simulation could not test abort scenarios because the brassboards were difficult to use. Exactly how the anomalies propagated is unclear because a bus undervoltage wiped out data from the recorder, nor could the anomalous behaviors be reproduced on ground.

During the emergency, the spacecraft fired its thrusters thousands of times. Fortunately, the fuel loss was tolerable because the thrusters were hardwired to fire only for fractions of a second. The mission was saved because the designers took precaution against fuel depletion during a software crash, a lesson learned from the previous failure.

Lessons learned
Create extensive, realistic nominal and anomalous operational scenarios for testing at every level, from unit through system test.

Implement robust simulators, including hardware-in-the-loop, for testing critical flight software functions.

Apply independent fault protection, such as hardware watchdogs, to mitigate risk in real-time systems, where errors can be so deeply buried as to be practically undetectable [6].

4.1.3.6.3 Maintain Lessons Learned

Any lessons learned from space vehicle anomalies or mishaps (loss of mission) should be captured and made available to future system designers.

5. Fault Management Systems Implementation Guidelines

The following paragraphs provide guidance for the implementation of an effective fault management system. These guidelines represent the current "best practices" performed in industry. Effective space vehicle fault management systems are integrated into the overall space vehicle CONOPs and design.

Elements of an effective fault management system include:

- Faults associated with the space vehicle are identified, assessed, documented, and the associated effects are either eliminated by design, or controlled by autonomous fault protection or ground-contingency operations. The methods for eliminating or controlling the fault or effect of the fault is documented and tracked throughout the program life cycle and for lessons-learned purposes.
- Identified faults include historical faults and lessons learned.
- Sound mission assurance practices are used to reduce the risk of common cause faults.
- Customer program office is kept abreast of the identified faults, fault mitigation, and protection methods used, and is included in key fault management design and verification decisions.

5.1 Fault Management System Development

This section describes suggested ways and techniques to *implement* a fault management system. This section is by no means the only way to implement a particular guideline as discussed in section 4, but can serve as a place to start as the techniques provided have been used successfully.

5.2 Fault Identification Methods

Faults may occur in any component of the system, or in the connections or interfaces between system components. As state earlier, these faults fall into two main categories:

- Random hardware failures, assumed to occur with a constant random failure rate.
- Common mode failures (including specification, design errors, improper parts and materials for the space environment, and workmanship errors).

Before choosing any method of failure detection, it is first necessary to identify the failures that must be detected and protected. Discussed below are the principal analysis techniques used today [1].

5.2.1 Failure Mode and Effects Analysis (FMEA)

Several definitions of failure mode and effects analysis are provided below.

"A procedure by which each potential failure mode or fault of a system is analyzed to determine the consequences or effects thereof on the system, to classify each potential failure mode according to its severity, and to recommend actions to eliminate, or compensate for, unacceptable effects." [14]

"A systematized group of activities intended to recognize, evaluate, and prioritize the potential failure of a product or process and its effects, and to identify actions that could eliminate or reduce the chance of the potential failure occurring, listed in the order of effect on the customer." [15]

"A procedure by which each potential failure mode in a system is analyzed to determine the results or effects thereof on the system and to classify each potential failure mode according to its severity." [16]

"A methodology to analyze and discover: (1) all potential failure modes of a system, (2) the effects these failures have on the system and (3) how to correct and or mitigate the failures or effects on the system. [The correction and mitigation is usually based on a ranking of the severity and probability of the failure]" [17]

An FMEA is a "bottom-up" analysis of a system design to determine the effect of component failures on the system. Basically the method looks at each component of the system. The analysis examines the failure modes of each component (and the failure rate) and assesses the effect of each failure mode on the operation of the system. This is a time-consuming process and it is necessary to show, from the FMEA, that a sufficient level of fault detection can be achieved to accomplish the system safety objectives. It is therefore very important that probable failures with a severe effect must be detected and appropriate recovery action must be taken, whereas it may be acceptable not to detect highly-improbable failures or failures that have no significant effect on system behavior. This analysis is not only time-consuming to perform, but also next to impossible to prove [18]. It is very difficult to ensure that all possible failure modes and where necessary, combinations of failure modes, have been considered and that their effects have been correctly analyzed [1]. Thus, it is not possible to prove that all single point failures in a system have been identified and removed.

The FMEA cannot be completed until the system design is completed. The FMEA needs to be started during the design phase so that actions can be taken early in the design process either to prevent the failure, or to minimize the chance of failures being identified that either cannot be detected or be effectively contained.

Performing a ranking of each failure mode's severity to the system and probability of occurrence is a failure modes, effects, and criticality analysis (FMECA). It is highly recommended that the fault management team trace each FMEA analysis item to a fault management failure ID (fault trigger). It is also highly recommended that the FMEA team concur with the fault management's team FMEA items to fault management trigger trace to ensure the correct interpretation of the FMEA has been made.

More information on FMEA guidelines and recommendations is available in the 2009 Mission Assurance Improvement Workshop Reliability/FMECA team document.

5.2.2 Fault Lists

Fault lists itemizing heritage or expected hardware faults can be used early in the design phase as a substitute for the FMEA. Once the FMEA is completed, the completeness of the fault lists should be to ensure all faults in the FMEA are included, or replace the fault lists with the FMEA to ensure complete fault coverage.

5.2.3 Fault Tree Analysis (FTA)

FTA is a top-down approach and can therefore be performed at the start of the design process and updated throughout the development and vehicle life cycle. The principle here is to take hazardous events, also known as root conditions, identified by a functional hazard analysis (FHA) and to analyze possible causes of those hazardous events [3]. For each potential failure event, a fault tree is created with the root representing the event, and the branches are the possible causes (including multiple failures) that create the event.

The NASA Fault Tree Handbook [19] provides a comprehensive description of fault tree analysis. As a deductive approach, FTA starts with an undesired event, such as failure of a loss or space vehicle power or loss of attitude control, and then determines (deduces) its causes using a systematic, backward-stepping process. In determining the causes, a FT is constructed as a logical illustration of the events and their relationships that are necessary and sufficient to result in the undesired event, or top event. The FT is a qualitative model that provides extremely useful information on the causes of the undesired event.

FTA can be simply described as an analytical technique, whereby an undesired state of the system is specified (usually a state that is critical from a safety or reliability standpoint), and the system is then analyzed in the context of its environment and operation to find all realistic ways in which the undesired event (top event) can occur. The fault tree itself is a graphic model of the various parallel and sequential combinations of faults that will result in the occurrence of the predefined undesired event. In some cases the FT contains Boolean logic to depict the failure combinations (ANDs and ORs). For unmanned satellite applications, it is not uncommon to see the Boolean logic eliminated and all combinations are assumed to be ORs, since these applications tend to only be required to be single-fault tolerant. The faults can be events that are associated with component hardware failures, human errors, software errors, or any other pertinent events which can lead to the undesired event. A fault tree thus depicts the logical interrelationships of basic events that lead to the undesired event, the top event of the fault tree. Figure 10 provides an example of an engineering fault tree.

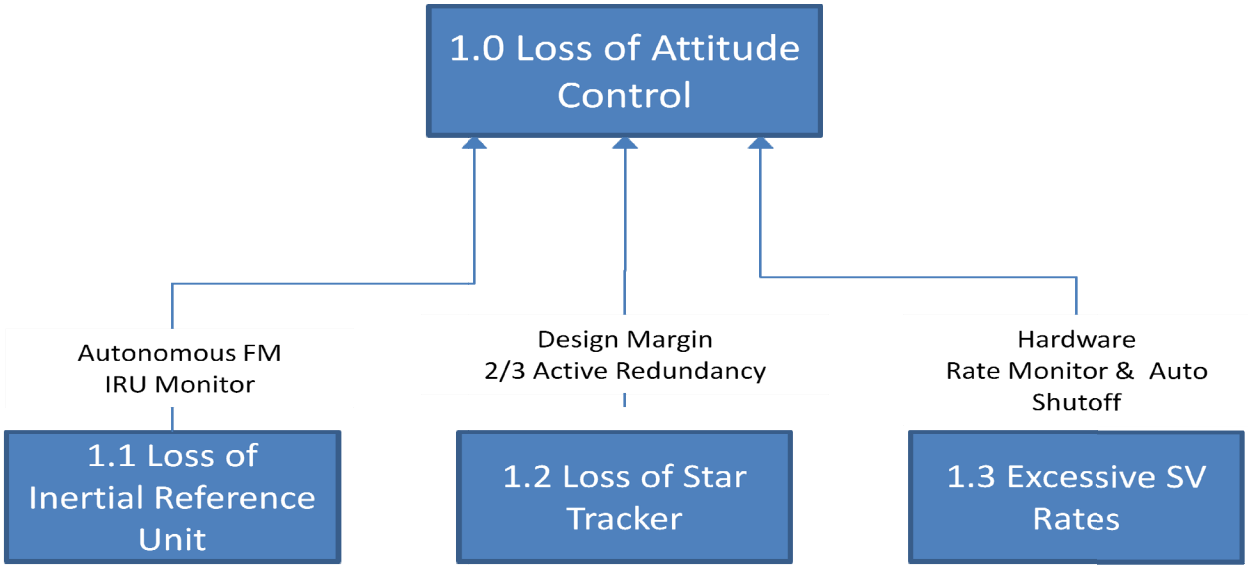


Figure 10. Fault tree example.

A FT can be transformed into its logical compliment, a success tree (ST) that shows the specific ways the undesired event can be prevented from occurring. The ST provides conditions that, if assured, guarantee that the undesired event will not occur. The ST is a valuable tool that provides equivalent information to the FT, but from a success viewpoint. Techniques for transforming the FT to its ST are described in the NASA Fault Tree Handbook [19] along with uses of the ST. A FT is more useful for FTrequirements development, design development, verification and coverage assessment than a ST.

The FTA is a useful method in defining the basic architecture and overall system design. It is also useful in identifying, at an early stage of the development cycle, the fault-detection and fault-response requirements arising from considerations of system safety or successful mission execution. Due to its top-down nature though, there is little hope of guaranteeing that every possible component failure mode, that

could cause or contribute to causing a hazardous event, has been considered. For this reason, it is normal to use the FTA approach to assist in the earlier stages of the design, and then to the use the FMEA to try to ensure that all types of failure have been considered. This combined FTA/FMEA approach has been widely applied, and is still essentially the approach recommended by the SAE "Guidelines for Certification of Highly-Integrated or Complex Aircraft Systems" [1].

The FTA should extend to the fidelity of the FMEA to better close the loop on failure modes and their propagation pathways (effects). In most cases the two analyses (FMEA and FTA) will not correspond as the bottom-up approach (FMEA) may have greater detail at the subsystem or card level than is necessary for sufficient fidelity in an FTA. Typical FTAs should extend to the level where fault protection is employed. FMEAs may extend to the card levels to include the interfaces, critical circuits such as redundancy switching and their failures, and to the individual IEEE components if necessary [19].

FTA can be applied to both an existing system and to a system that is being designed. When applied to an existing system, FTA can be used to identify weaknesses and to evaluate possible upgrades. It can also be used to monitor and predict behavior. Furthermore, FTA can be used to diagnose causes and potential corrective measures for an observed system failure [7].

It is important to understand that a fault tree is not a model of all possible system failures or all possible causes for system failure. A fault tree is tailored to its top event that corresponds to some particular system failure mode, and the fault tree then includes only those faults that contribute to this top event. Moreover, these faults are not exhaustive—they cover only the faults that are assessed to be realistic by the analyst. It is stressed that careful peer review of the fault tree is necessary.

The FTA has a wide variety of uses and roles it can play in fault management development and mission operations. The FTA can and should be used throughout the life cycle of the system from design through system implementation and improvement. As the system proceeds to the end of life, its performance can be monitored to identify trends before failure occurs. When consciously used to assist decision-making, the payoffs from FTA generally far outweigh the resources expended performing the analysis.

5.2.4 Function Based Fault Identification

System and subsystem block diagrams can be used to identify unit and interfaces that may fail. These block diagrams can then be used early in the space vehicle architecture phase to influence the vehicle or subsystem architectures to simplify fault management. The block diagrams are also useful in the development of fault lists and fault tree decomposition.

5.2.5 Single Event Effects and Criticality Analysis (SEECA)

If the effects of radiation induced effects cannot be eliminated by part selection, it is necessary to identify the effects of single event effects on the space vehicle. These effects can either be included as a failure mode in the FMEA or a separate analysis can be performed, similar to the FMEA to identify the effects of single-event effects. Such an analysis is subject to the same pitfalls and usage as the FMEA.

5.2.6 Byzantine Resilience (BR)

The FTA/FMEA approach requires extensive and detailed analysis, and it is argued that this analysis process is prone to error and is unlikely ever to fully succeed [19]. An alternative approach is to design the system to provide "Byzantine resilience" (BR). The BR approach consists of $3n + 1$ redundant channels, each identical, where n is the number of faults that must be tolerated. The principle of the BR

approach is that any deviation from expected behavior is regarded as a fault. Because deviation in performance can occur due to variations in component tolerances, wear, ageing, etc., can result in unexpected behavior, the BR approach is difficult to apply [1].

Many space vehicle components are subject to variations in performance, such as sensors, actuators, etc, and high levels of redundancy increasing the overall vehicle weight. Therefore, BR is not typically applied or applied in limited instances.

5.2.7 Analysis of Common Mode Failure (CMF)

The FMEA, FTA, and BR analysis techniques described prior deal fundamentally with the problem of random hardware failure of components. The space vehicle hazard analysis contained in section 2.2, shows that the cause of most space vehicle failures are common mode failure. Therefore, to obtain a robust space vehicle design it is necessary to consider these types of failures.

Failures due to external interference are generally dealt with by specific analyses and testing to show resilience to the various types of external threat [1]. For space vehicle, these specific analyses include EMI/EMC, radiation hardness, charging, and debris/micrometeroid analyses. Failures due to specification, design or implementation errors present a greater problem. FMEA will not reveal these types of failures and the BR approach is of no use if the redundant channels contain a common fault. FTA may be used in a limited manner, in that the FT for a particular event may include a contribution from this category of failure. A probability of the failure cannot be included in the analysis, but an objective or integrity target can be set and fault detection requirements can be identified. The BR approach though is inappropriate for detecting software failures because it relies on identical software running synchronously on identical redundant computing channels [1].

For space vehicle reliability calculations, software is usually considered to have a reliability of 1. However, despite continuing development and improvement in formal methods for software development processes and reliability analysis, any proof or assumption in software reliability does not take into account the possibility of errors in the original specification.

Dissimilarity between software (and hardware) used in redundant channels may be used to reduce the probability of malfunction due to design or implementation errors, but the benefit gained from this is difficult to quantify. The independence of dissimilar sets of software, produced from a common requirement or specification, has been questioned by many and has been shown to be unreliable [20]. To achieve greater independence, functional dissimilarity should be employed [1]. This, ideally, should use independently developed specifications for software, or hardware implementations for critical space vehicle functions (command, telemetry, battery charge control).

5.2.8 Analysis of Human Error

The analysis of the possibilities and effects of human error is often neglected in system design, or at least is not performed explicitly. It is often assumed that the operator will not give erroneous commands to the system, or that if (s)he does then it is not the fault of the system. Similarly, it is often assumed that the operator can be relied upon to take the correct and timely course of action to overcome the effects of failure elsewhere in the system. To some extent this approach has to be accepted. It will always be possible for the operator to cause an accident through malicious intent, gross negligence or incompetence [1].

The space vehicle design should implement some form of mitigation to address the risk from operator error. Human error risk in space vehicle systems are mitigated through the use of two-step commands for hazardous and critical commands, constraint checking, and the use of trained and certified operators.

5.2.9 System Out-Of-Tolerance Conditions

There has been a lot of discussion about the pieces of detection and responses needed to provide fault protection to a space vehicle. In addition to the many types of detection requirements, there is an additional set that are more general in nature. The space vehicle may exhibit behavior that does not point to a specific problem, but is more generic in nature, such as vehicle momentum out of tolerance, vehicle rate or attitude error beyond tolerance, etc., that require action to allow the vehicle to maintain control. The fault protection system should consider the use of out-of-tolerance conditions in the fault protection design.

5.3 Risk Assessment Methods

A probabilistic risk assessment (PRA), models sequences of events that should be performed to identify undesired end states occurrences and is described in the NASA Fault Tree Handbook [19]. FT are generally the work horses of a PRA, providing causes and probabilities for all the system failures involved, as well as a framework for quantification of the sequences.

As shown in section 2.2, most space vehicle failures are the result of a common mode failure. The causes for these common mode failures are usually the result of failures due to specification, design, or implementation errors. The FMEA will not reveal these types of failures and the byzantine resilience approach is of no use if the redundant channels contain a common fault. The FTA may be used in a limited manner, in that the FT for a particular event may include a contribution from this category of failure. A probability of the failure cannot be included in the analysis, but an objective or integrity target can be set and fault-detection requirements can be identified.

Because the PRA requires knowledge of the probability of the failure to produce reliable, quantitative results and the probability of most of the causes of space vehicle failures is not well known, PRAs are not recommended to assess the probability of mission success [19].

5.4 Fault Management Design Implementation Methods

The space vehicle architecture is derived from the mission functional, operational, performance, fault-protection requirements, and heritage architectures for the various space vehicle subsystems. Fault protection architectures all contain a fault-detection portion (monitors) and a response portion, but differ in the following areas:

- Hardware Redundancy: Block compared to unit/slice compared to functional redundancies compared to hybrid
- Autonomous fault protection architecture:
  - Sequence driven compared to set command responses
  - Table-driven compared to hard-coded compared to hybrid
  - Centralized compared to distributed compared to hybrid
- Fault Detection: Allocation of fault detection to hardware/software/procedures

- Fault Responses: Single thread compared to multi-thread compared to multiple levels
- Contingency Modes: Sun-pointed compared to earth-pointed compared to inertial

This section describes some commonly-used techniques for providing redundancy, fault detection, reconfiguration responses, contingency modes, safe mode, and low-power response mode.

5.4.1 Fail-Safe Compared to Fail-Operate

Fail-safe designs revert to a non-operating state upon detecting a fault. Only forms of passive survival modes are considered fail-safe states. For example, a vehicle that sheds loads, and relies on gravity gradient to establish a power-positive attitude and then waits for ground intervention to re-establish control is a form of fail-safe survival mode.

Fail-operate designs detect faults and reconfigure hardware and/or operating modes so that space vehicle attitude control, and power-thermal balance are maintained. The hardware reconfiguration can be a swap to a redundant unit or a swap to a redundant set of units needed to perform a function depending upon the architecture used. The transition to contingency operating mode can include ceasing mission operations and transition to a more benign attitude control mode, transitions to a sun-point mode or another power-positive mode. Hardware swapping may also occur as part of the mode transition. For fail-operate designs, it is important that all suspect equipment is swapped, including data paths. Otherwise the fault may persist and propagate.

5.4.2 Hardware Redundancy

The amount and type of hardware redundancy is primarily determined by the program reliability requirements. However, fault management concerns need to be considered when making the architecture trade studies, in that the redundancy scheme used drives the complexity of the fault management increasing the amount of testing and analysis required for verification. Likewise, the amount of required fault tolerance can drive the amount of redundancy needed.

5.4.2.1 Space Vehicle Level Redundancy

The brute-force replication of an entire system is referred to as system-level redundancy. Fault management implementation involves the isolation and removal of the entire system. An example of system-level redundancy would be a space vehicle, with a spare copy available to take over mission when commanded to do so.

Advantages:
System-level redundancy allows the development of less complex space vehicle, including single-string vehicles, reducing the individual vehicle cost.

Disadvantages:
System-level redundancy requires at least two vehicles be developed and launched. Feasibility is determined by launch costs.

5.4.2.2 Block Redundancy

Block redundancy replicates an entire functional block, as shown in Figure 11. For block-redundant functions, fault management involves the detection of faults and a transition to the redundant set of hardware.

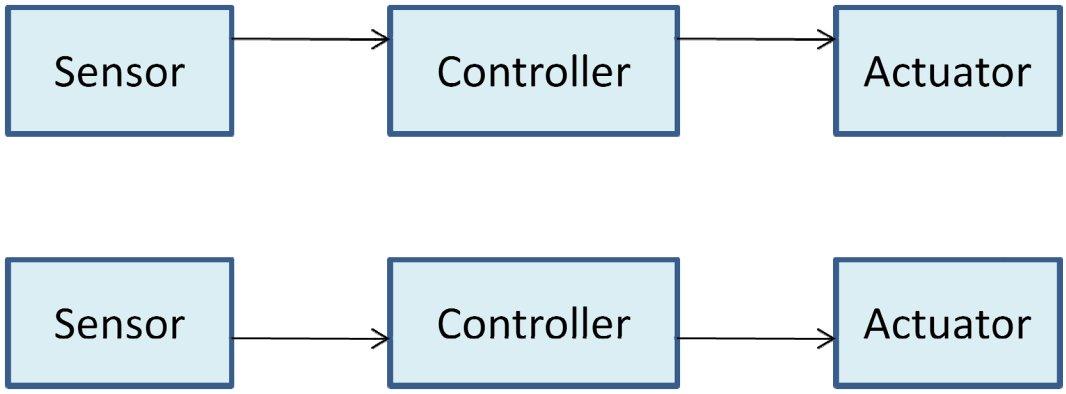


Figure 11. Block redundancy.

Advantages:
Block redundancy minimizes the possible number of fault responses, allowing for a single-fault response. The absence of cross-strapping minimizes the possibility of fault propagation.

Disadvantages:
The lack of cross-strapping does not allow for graceful degradation. Replication of the hardware does not provide any functional redundancy, allowing for design diversity to help mitigate common cause failures.

5.4.2.3 Unit or Slice Level Redundancy

Unit redundancy replicates each unit and units are interconnected (cross-strapped), as shown in Figure 12. Unit redundancy can be employed at either the box or slice level. For unit or slice-level redundancy, fault management operates to isolate and remove individual suspect units or slices or perform block switching. Redundant units can be either cold, hot, or warm spares.

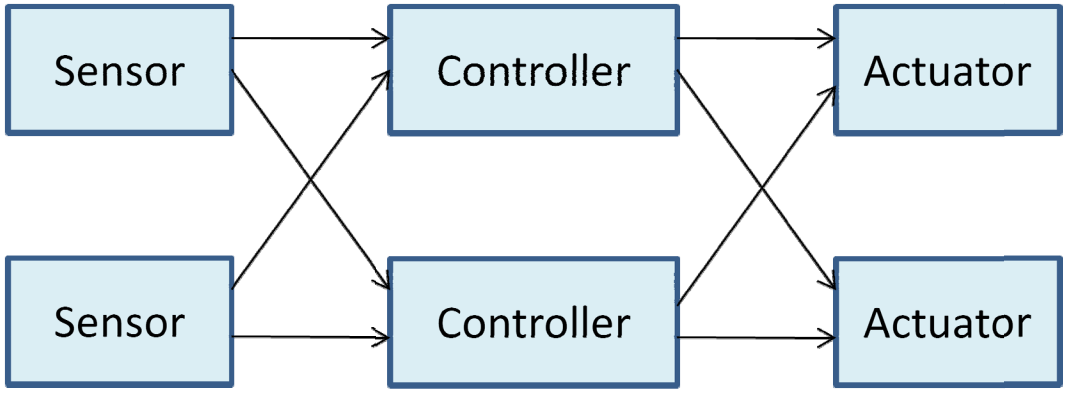


Figure 12. Unit redundancy.

Advantages:
Unit and/or slice redundancy, when cross-strapped increases reliability performance and provides for graceful degradation.

Unit redundancy also provides for the minimization of hardware swapping to the unit, compared to the entire string of units, for fault management responses. The swapping out of a faulty unit allows fault management to implement responses that allow continued mission operations.

Disadvantages:
Unit and/or slice redundancy increases the number of paths that must be tested. Care must be taken to prevent single points of failure in the interconnections as well as fault propagation. Additional effort is required to analyze these interconnections for failures.

The increased number of paths and fault management responses, requires additional analysis and testing to ensure that the correct response runs to reconfigure the suspect hardware and perform mode transitions.

5.4.2.4 Hybrid Redundancy

Hybrid redundancy is a combination of the above types of hardware redundancy schemes. The use of hybrid redundancy allows the advantages for the various redundancy schemes to be realized while controlling the disadvantages. Most space vehicle employ some amount of hybrid redundancy. One example of hybrid redundancy is a non-redundant sensor that is connected to redundant processing equipment, such as an earth sensor that is used only after a fault has occurred (the earth sensor is not needed for nominal operations). Other examples of hybrid redundancy is active redundant battery charge control; passive, redundant processor with block redundant for sun-sensor and thruster control; cross-strapped processor and star trackers; and functional redundancy that uses thrusters in place of reaction wheels.

5.4.2.5 Triple-Modular Redundancy

Triple-modular redundancy (TMP) involves using three units or functions and using voting logic to reject the output of one if it differs from the other two. Triple modular redundancy is not the panacea the words suggest. It is important that the execution of the circuits on the PWB and component placement do not defeat the intent of the triple voting. The connector pinout, the signal trace routing (both within a layer and layer-to-layer), component placement, and component failure modes need to be critically reviewed in detail to ensure no single failure exists that compromises the functions that are triple voted. This is critical as there is no redundancy to switch to should a fault that compromises outputs to two functions develop in the voter circuit. It is also highly recommended that the TMR circuit contain test points to allow verification of the circuits components at a high-level of integration. The TMR circuit's properties are such that they mask out a failure in the circuit, so verification that all of the components are properly functioning is key to ensuring that the space vehicle is not one failure away from mission failure. The scrubbing of SEU's upstream of the voter circuit also needs to be thought through. While 1 SEU to a logic block upstream of the voter circuit can be tolerated, without a mechanism to flush the SEUs from the logic blocks, accumulated SEUs in different logic blocks over the lifetime of the function may result in faulty logic block outputs (more than 1). Proper verification that all of the components are properly functioning prior to launch, is key to obtaining the benefits of TMR.

Advantages: TMR increases the unit's reliability calculation.

Disadvantages: TMR can introduce single faults in the voting logic, requiring careful layout design and review. In addition, test points need to be provided to ensure that a failure has not already occurred prior to launch.

5.4.2.6 Functional Redundancy

Functional redundancy provides independent functional backups to units and/or space vehicle functions, providing for some level of design diversity to help protect against common-cause failures.

Advantages: Functional redundancy provides protection for select common cause faults and provides for graceful degradation as components fail.

Disadvantages: The addition of functional redundancy into the space vehicle architecture adds complexity to the hardware design, software, and the fault-management design, increasing the amount of analysis and testing to be performed. Functional redundancy also uses weight, power, command, and telemetry resources.

Examples of functional redundancy are:

- Independent Attitude Sensors—Sun sensors, earth sensors, and magnetometers are viable backup sensors to star trackers.
- Independent Actuators—Thrusters or magnetic torquers are viable backups to reaction wheels or control moment gyroscopes.
- Independent Safe Mode Processor—Safe mode or Contingency modes implemented in an independent processor provides a viable backup to redundant processor, allowing a safe state to be obtained in the event of a processor fault.
- Ground Control without Computer—The ability to control attitude, battery charge, and heaters via direct ground commanding provides an alternate backup to independent processors in the event a fault results in loss of control. This backup method is only viable for space vehicle that have sufficient ground visibility to keep the space vehicle in a recoverable state.
- Hardware Implemented Battery Charge Control—Hardware implemented battery charge control provides a viable backup for software implemented battery charge control, should a common-cause processor problem occur. Power should always be safe without the flight processor (although degraded battery charging efficiency is allowable).
- Command Links—Alternate command paths, such as Omni antennas or S-Band, provide a viable backup path to the primary command link.
- Telemetry Links—Alternate telemetry links provide a viable backup to primary telemetry links.
- Thermo-Static Controlled Heaters—Thermo-static controlled heaters provide a viable backup to software controlled heaters, providing protection in the event of a processor fault.

Advantages:
Functional redundancy provides protection for select common-cause faults and provides for graceful degradation as components fail.

Disadvantages:
The addition of functional redundancy into the space vehicle architecture adds complexity to the hardware design, SW, and the fault management design, increasing the amount of analysis and testing to be performed. Functional redundancy also uses weight, power, command, and telemetry resources.

An example of the advantage of functional redundancy is on the Alexis space vehicle. A solar panel broke loose during launch, damaging the magnetometer and causing the satellite to tumble. Fortunately, the spacecraft was able to use Earthshine to trickle charge permitting ground controllers to regain control six weeks after launch. Video onboard the Pegasus launch vehicle revealed that recovery was possible despite

the damage. Several weeks later, a downlink indicated that the satellite was able to charge partially up using Earthshine. By controlling the magnetic torquers from the ground, operators were able to determine the satellite's orientation and begin on-orbit operations three months after launch [6].

5.4.2.7 Software Redundancy

5.4.2.7.1 Backup Software

Backup software provides a viable means of space vehicle control in the event of a primary software failure. Backup software typically resides in the same processor and consists of the basic functions needed for safe space vehicle control in a safe but degraded space vehicle operation.

Advantages:
Backup software provides protection for software errors or computational errors from executing mission operations.

Disadvantages:
The use of backup software adds hardware components and increases the complexity of the processor design.

5.4.2.7.2 Multiple Copies of Flight Program

Depending on the type of devices used to store flight software used in an assembly, it may be necessary to store multiple copies in the non-volatile storage to corruption of the flight software image. Some non-volatile storage devices may have hazardous implementation characteristics [17]. If the flight software image is critical to implementing the on-board fault management, the fault management team should consider the usage of multiple copies of the flight software and a mechanism to determine what conditions the copies would be used to ensure fault management is able to execute.

5.4.3 Integrated Modular Avionics (IMA)

New, integrated avionics architectures are being applied to space vehicle designs. The aim is to reduce the costs associated with conventional avionic architectures. The main features of these architectures are listed below:

- The boundaries between subsystems are less distinct. Different systems may use common resources.
- The 'black-boxes' are replaced by 'line replaceable modules' (LRMs). These modules are not necessarily dedicated to a particular function. The module may be utilized by several functions (e.g., a power-supply module).
- Sensors, actuators, etc., are connected to the modules via data buses. Interfacing of the sensors and actuators to the data bus is achieved by localized electronics. Thus electronics and processing facilities are distributed rather than centralized.
- Communication between modules is achieved via high-speed, parallel data buses.
- All modules are of standard types. It is also intended to make extensive re-use of SW or HW components.

This approach is known as integrated modular avionics (IMA) [1]. The introduction of IMA into satellite HW design has a significant impact on the fault management problem.

5.4.3.1 Effects of IMA on Existing Fault Management Techniques

The most significant distinctions between conventional and IMA systems architectures are that the boundaries between different subsystems are less clearly defined with IMA, and that the use of standard modules greatly increases the potential for common mode failures affecting many system functions [1]. These differences complicate the fault management requirement allocation process, fault identification, and fault coverage analysis, using the existing methods. It also raises the need for common mode failure analysis and possible protection for critical space vehicle health and safety functions (telemetry, command, power, and attitude control). The SAE certification guidelines clearly recognize this, and require the aircraft manufacturer to perform common-cause fault analyses starting with the identification of 'aircraft-level functions' [1]. The effect on satellite failure rates due to common-cause faults has yet to be determined.

In IMA there is a distinction between the central IMA processing resource and the other components of the system. The interfaces between the central processing resource and the other system components are greatly simplified with IMA, due to the replacement of dedicated wiring carrying an assortment of signal types, by data bus communications [1]. This somewhat allows the fault management problem to be split in two: one part considering the central processing resource and data bus communications, the other considering the other components of the system.

The use of data bus communication within a system offers potential improvements in fault diagnostic capability. Faults in other system components can be detected locally and reported back via the data bus. Faults in data bus communications can be readily diagnosed by monitoring data refreshment, parity, etc. Such capability increases the scope and complexity of the fault monitoring and fault responses. There is, nevertheless, an overall benefit because analyses can be performed at a component, rather than a system level, and the problems associated with the detection and isolation of wiring failures are largely removed, allowing less components to be reconfigured to isolate a fault.

With IMA the results of the FMEA or FTA for a particular function is no longer be able to define the subsystem fault management requirements, since functional elements are not necessarily be dedicated to that subsystem. The failure modes from each function need to be analyzed together to determine the fault management requirements. This analysis is particularly important if autonomous hardware reconfiguration is needed to preserve critical functions. It requires additional coordination between the designers of the different subsystems and the fault management designer so that all of their requirements can be integrated. Great care must be taken to avoid the introduction of new, and possibly devastating, common mode failures.

The requirements for other components and/or subsystem of the system, and the monitoring requirements for these components can be addressed using the existing analysis techniques. The FTA is particularly useful for examining failures at the functional level and for integrating all of the different subsystems and levels of fault protection and autonomous fault management. Use of the FTA allows the fault management designer to identify potential sources of fault response interference and to identify critical resources that need to be reconfigured quickly to maintain operation of critical space vehicle functions.

Examples: Computing and databus resources typically need to be reconfigured quickly so as to be able to maintain attitude control, battery charge control, and thermal control.

5.4.4 Autonomous Fault Protection Implementation Methods

5.4.4.1 Autonomous Fault Protection Architectures

There are a variety of viable autonomous fault protection architecture schemes. The choice of the on-board fault protection architecture depends upon the coding language and overall software architecture.

Fault protection consists of fault monitors, which detect faults, and fault responses, which then reconfigure the space vehicle to contain the fault. Fault protection may be implemented on-board or in ground procedures, depending on space vehicle autonomy requirements. On-board fault protection is required for rapidly developing faults, or for faults which interfere with the ground's ability to communicate with the space vehicle. Other faults may be addressed by ground-based tests (alarms) and responses (contingency procedures).

Fault detection algorithms can be either hard-coded, table driven, or a combination of both (hybrid). Fault response algorithms can be hard-coded responses, stored command sequences, table driven sequences, macros, or combination of these techniques.

Hard-Coded Algorithms—Hard-coded algorithms allow for computational algorithms to be used to determine of a failure has occurred.

Advantages:
Hard-coded detection and response algorithms are integral to the flight code and are verified in the same manner as the flight code. These algorithms can be modified by loading new flight code into re-programmable processors.

Disadvantages:
Because hard-coded algorithms are part of the flight code, they need to be defined, implemented and verified as part of the flight code delivery. These algorithms are typically defined late in the flight SW development cycle.

Table/Database Architectures—Table driven architectures use a database defining the parameters to be monitored, failure thresholds, persistence values, etc., and the associated responses for each monitor, allowing easy modification.

Advantages:
Table or database driven fault detection and responses allow for easy modification after the code has been qualified.

Disadvantages:
Correct implementation of the fault management monitor and response database is essential to safe fault management design. Good configuration control of the database, including complete verification and validation testing are still required for all database changes.

Algorithm Location—The location of the detection algorithms and responses can be centralized in the primary processor or in a software code unit, or distributed amongst software code units or HW units. The activation of responses can be sequential or parallel, again depending upon coding language, SW architecture and designer preference.

Sequential Responses—Sequential responses only allow one fault response to be active at a time. The first response to trigger executes and completes before a second response can execute. For sequential

response architectures, it is important that the analysis be performed to ensure that the response that executes first clears the fault, or at least does not cause the fault condition to propagate to further aggravate the anomaly or damage hardware.

Parallel Responses—Parallel responses allow multiple responses to execute at the same time. For parallel response architectures, it is important that the responses due not interfere with each other and that a safe vehicle state always be reached.

There are two basic fault protection architectures: centralized and distributed.

Centralized Fault Protection Architecture - Centralized fault protection architectures, shown in Figure 13, have the fault detection monitors and fault responses located in the primary processor or a single software code unit.

Advantages: Centralized architectures allow for the use of table-driven monitors and/or responses. Fault protection verification activities are concentrated to a single-implementing subsystem.

Disadvantages: Fault detection and responses may be implemented in units removed from the source of the fault, potentially introducing additional failure paths.

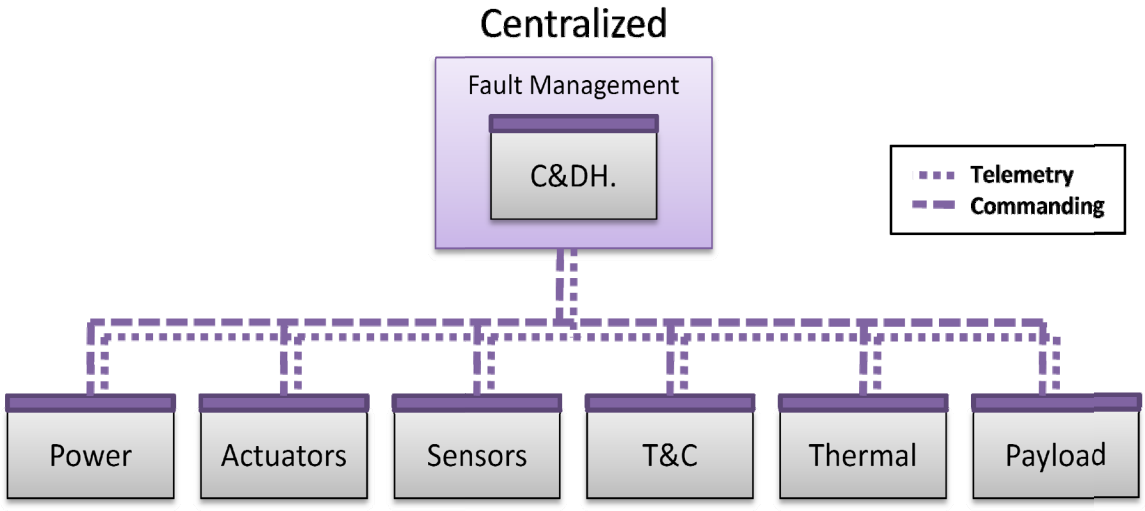


Figure 13. Centralized fault protection architecture.

Distributed Fault Protection Architecture—Distributed fault protection architectures, shown in Figure 14, have fault detection monitors distributed amongst software code units or hardware units.

Advantages: Distributed architectures allow the fault monitors or fault-detection algorithms to be located more closely to the source of the potential failure.

Disadvantages: Fault protection verification activities are distributed amongst the implementing subsystems, increasing verification complexity.

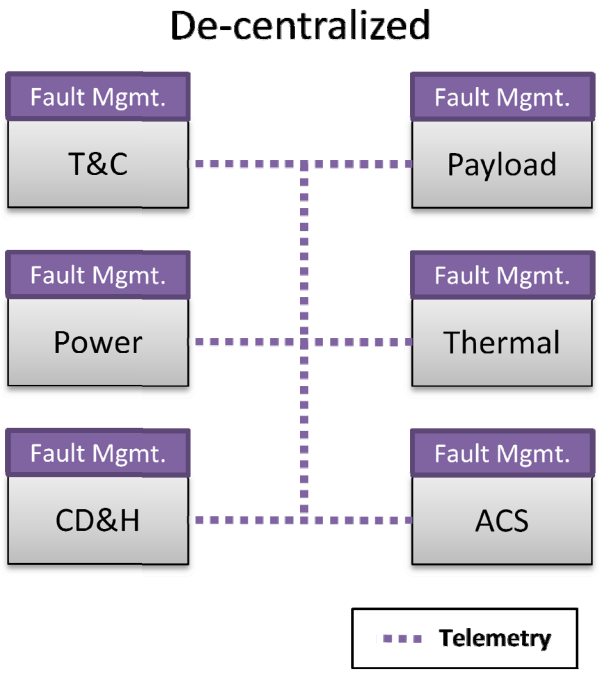


Figure 14. De-centralized fault protection architecture.

Regardless of the autonomous fault-protection architecture selected, the fault-protection guidelines listed in section 4 should be followed.

5.4.4.2 Fault Detection Methods

Fault monitors are used to perform fault detection. Fault monitors should include provisions for minimizing the likelihood of spurious fault indications—provisions termed spurious-signal negation. These provisions can include filtering, persistency, or cross-checking. Fault monitors should also include provisions to resolve ambiguous or common-mode faults—those which could trip multiple fault monitors. Ideally, the space vehicle design will make information available to the fault monitors which allow them to be independent. When this is not possible, the fault monitors need to be prioritized, or the associated fault responses need to include some embedded fault management to avoid having the fault responses interfere.

Various techniques for implementing fault monitors are described in the following text.

5.4.4.2.1 Sensor Faults

Commonly used sensor fault detection techniques are described below. When detecting sensor failures, care must be taken in determining that a specific sensor has failed. The source of the fault could be in the sensor, the sensor data transmission, or the receipt and processing of the sensor data. Sensor-failure fault responses must reconfigure all suspect failure source equipment, including data transmission paths to prevent the failure from propagating.

Sensor Built-in Test (BIT): For complex sensors, the sensor built-in test data can be used to detect a sensor failure. Sensor BIT is a reliable means to clearly identify that the specific sensor has a failure condition.

Examples: Many star trackers and inertial reference units are processor based and have their own fault-detection algorithms and issue error flags to indicate a detected failure.

Thresholds—Direct measurements or errors between actual and commanded values are compared to established positive and negative thresholds. Care must be taken when using thresholds for failure detection. If the thresholds are set too wide, the effect of the anomalous behavior may not be contained and the failure is allowed to cause hardware damage or prevent the ability to establish a recoverable space vehicle state. If the thresholds are set too tight, false fault response trips can occur, causing unnecessary hardware reconfiguration or mission outages. Persistence counters are used to aid in the prevention of false or premature execution of the fault responses. There are two approaches that can be taken to determine the optimum detection thresholds and filtering or fault confirmation times. The first approach is to use the FMEA to identify the actual failure modes and to design the monitor to detect these specific failures. The second approach is to use the FHA and FTA to identify system-level failure events and to design the monitors to protect the system from the feared events [1]. Thorough analysis and testing are required to ensure that the thresholds and persistence counters are set correctly, see section 4.1.3.5.5 (ITP).

Thresholds settings can vary over life as component performance degrades or vary with space vehicle configuration, or with attitude control modes. The use of monitor threshold requires the ability to adjust the threshold and persistence counters by ground (command or uploading database values), sequence command, or by fault protection responses as appropriate.

Examples: Voltages and/or currents that are out of range. Temperatures that are either too hot or too cold. space vehicle rates that are too high.

State Estimator—State estimators use the commanded state and knowledge of the system behavior to compute the expected estimated state and compare the estimated state with sensor data. A failure is assumed if the difference between the estimated state and the sensor data exceeds a threshold. Analysis and test with real sensor outputs is needed to ensure that the state estimator correctly detects sensor faults.

Examples: Sensed actuator state does not agree with the commanded actuator state. Sensed space vehicle rates and or attitude does not agree with commanded rates or attitude.

Reasonableness Tests – Some sensors report engineering values that are clearly inconsistent with the physical process when they fail. These reasonableness values can be established a-priori and be used to detect failed sensors inputs. This approach does not work for sensor failures that produce a constant or offset value [11].

Examples: Temperature sensors output “00” or “FF” are considered failed.

Analytical Redundancy—The use of analytical redundancy compares sensor outputs from different sensors and uses physical properties to compare sensor data through a common property [11].

Example: Using battery pressure, current monitors, temperature, and voltage to determine battery state of charge.

Redundant Sensors—For detections where an inadvertent trip would result in a severe response (For example, downlink loss, irreversible hardware swaps, large use of expendables, critical sequence cancellation), and where a sensor anomaly could cause an inadvertent trip, independent physically or functionally redundant detections are employed such that simultaneous detections are necessary for response initiation [7].

Examples: Using sun sensors to detect stellar inertial determined space vehicle attitude errors.

5.4.4.2.2 Actuator Faults

Commonly-used actuator fault detection techniques are described below. When detecting actuator faults, care must be taken in determining which specific actuator has failed. The source of the fault could be in the actuator command, actuator command output, data transmission path, actuator processor, or the actual actuator. It is not always possible to isolate actuator failures from sensor failures, data transmission failures, processor failures, or input/output electronics failures. Also, the effects of actuator failures occurs after the failure has occurred, requiring a rapid response to limit the effect of the failure and restore the space vehicle to a controlled state. Thorough analysis and test is required to ensure proper threshold and persistence settings and that the correct response runs if multiple response system is used. In addition, the structure must be designed to withstand the effects of an actuator failure. Therefore, both actuator and sensor components of the fault signature/path need to be addressed.

Actuator Built-in Test: For complex actuators, the actuator control electronics BIT data can be used to detect an actuator control failure. Actuator BIT is a reliable means to clearly identify that the specific actuator control electronics or actuator has failed.

Examples: Some control moment gyroscopes have control electronics with BIT provided.

Wraparound Tests—Actuator wraparound tests compare the sensed actuator output with the commanded value. If the sensed effect does not match the commanded state, then a failure is assumed to have occurred.
Examples: Resolvers are used to measure actuator position and actual resolver position is compared to commanded resolver position. Rate and attitude sensors are used to detect the effects of commanded thruster or reaction wheel commands to change space vehicle rates and/or attitude.

5.4.4.2.3 Single-Event Effect Faults

There are many instances where a single-event effect (or single-event upset) can cause what looks like a failure signature to the fault management system. The fault management system must be able to address instances of single-event effects. The three most common single-event effect items for the fault management system to address are single bit changes in hardware storage elements (registers, state machine flip-flops, etc.), single-bit error in memory, and double-bit errors (in memory).

Single-bit errors in registers tend to affect the ability of a sensor or actuator to function or to communicate. In most cases, the lack of a functioning sensor or actuator falls into a fault trigger that has already been defined. In the case of the data transmission being affected by the single-event effect, the detection of the fault may take some time. To aid in detection of single-event effects on the data paths, the fault management system should implement periodic data path checks to assess communication paths.

Single-bit errors in memory can take on many forms. For many applications, a single-bit error is a transient that is overwritten in the next processor cycle. For data that is not overwritten for some period of time (static data), the data should be protected by a minimum of a parity check. The fault management system needs to know if static data has been changed before it is used as it may be some time before the data is overwritten with valid information. Another recommended approach to single-bit errors in memory is the use of error detection and correction (EDAC). This allows the detection of the single bit error and the correction so that the next time the data is used the single-bit error has been corrected.

Double-bit errors are a low-probability occurrence that the fault management must still address. The effects of a double-bit error can be devastating to a space vehicle. The fault management system should be able to detect a double-bit error and respond without using information from the memory that contains the double-bit error. In practice, this usually requires a swap of the affected equipment (generally a processor).

There may be instances where the implementation of a data path check, parity, or EDAC is not feasible. An alternative method to address single-event effects is to perform a periodic refresh of registers, constants, static data, etc. from a protected source [like a programmed read-only memory (PROM)]. One technique that has been used successfully is to perform a periodic reset of a unit (places all hardware registers in a known state) during a defined time of inactivity (so the reset is transparent) and re-write all of the required register locations from a protected source.

5.4.4.2.4 Data Transmission Faults

Data busses used for the transmission of commands and telemetry data to sensors, actuators, other processor units, etc., require techniques for detecting data communication faults. Many of the standard data buses were designed with fault detection in mind, such as the MIL-STD-1553B data bus [23]. Custom data buses must also provide fault detection capabilities. It should be noted that not just the external databus needs to be checked. If the datapath that is being checked includes a path internal to the unit (like a PCI bus), the internal bus integrity needs to be checked. Commonly used data communication fault detection techniques are described below:

5.4.4.2.4.1 Parity Check

A parity generator/checker adds a bit called a parity bit to a word of data such that the total number of “1”s contained in both the word of bits and the parity bit is either an even or odd number. The choice of even or odd parity is determined by the designer in advance.

5.4.4.2.4.2 Checksums

Checksums are used to detect failures in blocks of data. The checksum is transmitted with the block of data, and a failure is detected if the received checksum differs from the transmitted checksum.

5.4.4.2.4.3 Cyclic Redundancy Codes (CRC)

A CRC is used in the transmission of blocks of data and are transmitted as part of the data block as a checksum. The receiving end performs a CRC on the block and compares the checksum to determine data integrity.

A cyclic redundancy check is a type of function that takes as input a data stream of any length, and produces as output a value of a certain space, commonly a 32-bit integer. The term CRC denotes either the function or the function’s output. A CRC can be used as a checksum to detect accidental alteration of data during transmission or storage.

5.4.4.2.4.4 Timeouts

A failure is declared if data is not received in the expected time interval.

5.4.4.2.4.5 Buffer Overflows or Underflows

A failure is declared if either too much or too little data is received.

5.4.4.2.4.6 Bit patterns within the data

If a known value is used and compared against the received value and fails, the datapath is suspect. Example: Examination of a space vehicle fault management fault detection found that the internal databus of the command and data handling (CDH) unit did not check the internal datapath as part of the data integrity checks. The internal bus check that was implemented included a heart beat signal, but the action of writing the heart beat register within the unit was sufficient to declare a good heart beat. The data values used as part of the heart beat write were treated as don't care bits. The bus was a bus which did not implement a parity check. Checksums, timeouts, nor bit patterns were used to assess data integrity. The analysis found that a shorting of bus data lines could result in corrupted data (ACS, EPS, etc) that the fault management would not flag as suspect (a valid heartbeat would prevent the unit from swapping). It was also determined that the ACS and EPS fault management algorithms did not contain a response to swap the CDH unit should the data be suspect (sensor and actuators were swapped, but not the unit that *processed* the data). This system left a portion of the datapath unchecked and could result in an uncontrolled vehicle due to a single fault [21]. Figure 15 illustrates the CDH, ACS, and EPS data boundaries within a CDH unit.

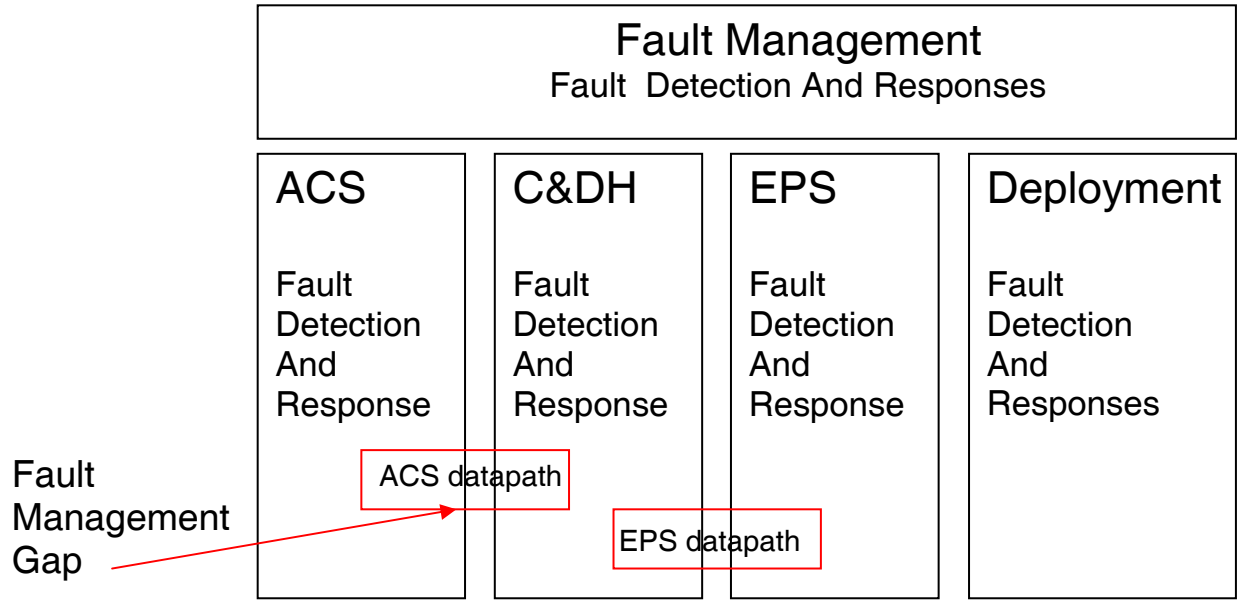


Figure 15 . Fault Management datapath boundaries.

5.4.4.2.4.7 Data Wraparound Checks

A data path is checked by sending a known value or bit pattern and then comparing it to the received value.

5.4.4.2.5 Power Faults

Commonly used power-supply fault detection techniques are described below. When detecting power-supply faults it may not be possible to detect a power-supply failure from another hardware failure. The

electronics should be designed such that power-supply failures do not result in anomalous voltages, currents, or logic states that can propagate to other units or functions.

Under-voltage Over-voltage Protection—Under-voltage and over-voltage detectors are used in units to ensure that a unit shuts off or goes to a safe state should the voltage supply fail outside the designed operating range.

Voltage Monitoring—For designs using external power supplies, voltages must be monitored for correct values to detect a power-supply failure and estimated state-of-charge (accumulated current sensing).

Estimated State of Charge—Accumulated current sensing and/or battery voltage, pressure, and temperature can be used to estimate the battery state-of-charge.

5.4.4.2.6 Thermal Control Failures

Wrap-around tests can be used to detect thermal-control faults. The sensed temperature can be compared with the expected value range. If the temperature is too hot, it can be assumed that a heater has failed on. If the temperature is too cold, it can be assumed that a heater has failed off. Many times thermistor reasonable tests are used to distinguish between thermistor faults and data-path faults. Redundant thermistors and heaters with staggered setpoints are also used to provide a passive response to single-thermistor or heater failures.

5.4.4.2.7 Attitude Control Failures

Many of the sensor and actuator fault-detection methods can be used or combined to detect attitude-control failures. Out-of-tolerance checks should also be used to detect anomalous attitude-control performance, such an anomalous momentum state, failure to point to the commanded attitude, or anomalous vehicle rates. Out-of-tolerance checks must have appropriate thresholds and persistence counters to be effective

5.4.4.2.8 Operator Error

Commonly used techniques are: Parameter range checking, constraint checking, and two-step commanding for critical events.

5.4.4.2.9 Processor Faults

The space vehicle, including the processor, must be designed to withstand the space environment and appropriate design measures taken to protect against the effects of environmental events.

5.4.4.2.9.1 I/O Failures

Sensor Input Failures—Sensor-failure detection monitors will also detect sensor input failures if both types of failures produce the same observables. It is important to make sure that redundant sensor-input processing is provided for redundant sensors. It is also important to consider the *entire* path (not just the sensor or actuator) when designing input/output (I/O) path checks and responses.

I/O is used to move data from one place to another. A key function of the fault management system is to detect when data is not being transferred as expected as an indicator of a fault.

Example: A sun-sensor failure producing zero voltage is indistinguishable from a sun-sensor input electronics failure.

Actuator Output Failures—The data wrap around test for actuator failures will also detect actuator-output processing failures. It is important to make sure that redundant actuator-output processing is provided for redundant actuators.

Example: A failed thruster (zero output) will produce the same symptoms as output electronics failure.

Data Transmission Failures—For fault management to be able to detect a problem with the I/O paths, something must be known about the I/O path that is deterministic. Deterministic properties include data-packet size, packet expected arrival time, or expected data content in the packet (bit pattern, checksum, or parity). These properties can then be used to construct a test of the I/O to determine its integrity.

Example: The computer is expecting a packet from a sensor of 25 words. The computer receives 16 words. Persistent incorrect packet size should result in the integrity of the entire path as suspect (including the destination).

Example: The computer expects data from a sensor by a predetermined time (within the processing cycle). The pre-determined time is generally determined by a timer. If the timer expires, the data path (including the destination) is suspect.

Example: As a practical I/O check, a known pattern is sent to another unit. A request for the known pattern is made and compared to determine if the I/O path is viable. Alternate ways of doing this include the use of checksums or parity.

5.4.4.2.9.2 CPU and Memory Failures

Hardware failures in the central processor unit (CPU) and memory devices can cause the software to stop functioning correctly. Thus, software alone cannot be depended upon to detect CPU and memory hardware failures. A two-pronged approach is needed for protection against potential CPU and memory failures. The first approach assumes that the CPU or memory device has failed with the result that the software ceases to function. The second approach assumes that the software still continues to function in spite of faults within the CPU and memory devices [11].

5.4.4.2.9.3 Watchdog Timers (WDT)

If there is a complete software failure, the CPU can be expected to cease outputting data to the I/O modules, or any data transmission paths. A watchdog timer makes use of this failure effect by having normally functioning software send out a continuously varying (WDT) status signal to a special hardware circuit. If the varying status signal stops, the WDT will wait for a period of time, called the timeout period, and then change its output to signal a failure. Some CPUs come with WDT built-in, which are primarily there to detect those software failures resulting in a program execution halt. This WDT cannot be depended upon to fully cover all CPU hardware failures and should be supplemented with an external WDT circuit as described above [11]. The monitoring of the WDT input status signal should be on a separate board, powered separately from the processor issuing the WDT. It is also recommended that the generation of the status signal from the processor be initiated from the processor processing cycle as a synchronous activity. This technique allows the synchronous system response to the WDT counter timing out (since the WDT resets are synchronous to the system). This also results in a deterministic response time within the system.

5.4.4.2.9.4 CPU Self-Tests

CPU self-tests supplement, but do not replace, the WDT. CPU tests run continuously and usually result in an interrupt indicating a fault condition has been detected. The faulty condition can be either transient in nature due to radiation effects or permanent. To prevent hardware errors in registers and memory data and address interfaces that do not result in a failure of the software to run, some type of protection must be provided. Software should provide a response for all detectable processor exceptions in the event that the failure does not result in an immediate WDT trip.

5.4.4.2.9.4.1 Memory Tests

The objective of real-time memory tests is to uncover and possibly correct memory faults before they surface as memory failures or anomalous space vehicle behavior. If a faulty *instruction* is not detected and then executed, a random program will probably begin executing, most likely leading to a WDT timeout. If a faulty data is not detected and used during execution, incorrect CPU output may occur. For the case of faulty *data*, it is difficult to distinguish between hardware and software errors. For example, incorrect data can be caused by incorrect data in software or a stuck-bit in hardware memory. Faulty data will need to be detected by one of the previous external tests [11]. EDAC or single-bit error correction, double-bit error detection (SECDED) are commonly used for memory fault protection. Special care must be taken when double-bit errors are detected. To isolate the processor executing a possible modified instruction or data, the processor should not be part of the response to a processor double-bit error.

Rationale: The processor may execute an instruction that is not correct due to lag time between instruction execution and notification of a double-bit error and cannot be trusted to perform properly.

5.4.4.2.9.4.2 Electronic Faults

Electronic faults, such as open and short circuits in components, connectors, and will result in functional faults that should be detected by the tests described above [11].

5.4.4.2.9.4.3 Clock Faults

Clock stoppage is detected by the WDT. Faults leading to significant increases in clock frequency will eventually result in CPU or memory access faults, leading to a WDT timeout. Small changes in clock frequency can alter attitude-control performance and possibly induce control-loop instability causing a serious anomalous behavior. Anomalous control-loop behavior must be detected by the tests described above with a response that triggers a CPU failure.

5.4.4.2.9.4.4 Software Faults

Simple software routines to detect processing timeouts or other unexpected software results are used to detect software faults (logic verification, or validation escapes.)

5.4.4.3 Fault Response Methods

Fault responses reconfigure the space vehicle to clear the condition, which triggered the associated fault monitor. Once a fault has been detected, fault responses can either reconfigure the software to disregard faulty sensors, to not use suspect actuators, reconfigure to redundant hardware, or to reconfigure the space vehicle to a contingency or safe mode to establish a safe state.

Fault responses should leave the space vehicle in a state consistent with ground contingency procedures. The space vehicle should be safe for at least a pre-determined amount of time. The space vehicle dynamic state should be consistent with the initial conditions assumed by ground procedures.
Fault responses need to be designed with the following priorities:

1. Protect critical space vehicle functionality, including the payload,
2. Protect space vehicle performance and consumables,
3. Minimize disruptions to normal mission operations, and
4. Simplify ground recovery response, including providing for downlink telemetry [1].

As described in the fault protection architectures, responses can be (1) single thread, with one response that reconfigures the space vehicle to transfer from primary equipment to all redundant equipment, (2) multi-thread, with multiple responses that reconfigure the space vehicle hardware, or (3) multiple levels, with multiple responses executed in order of increasing severity and/or urgency.

In cases where several conditions may trigger the same fault monitor (absent the availability of any discriminating information), the fault response may need to execute in stages, each stage addressing one of the possible fault conditions—an incremental approach. Alternately, the fault response can address all of the possible fault conditions at once—a block-swap approach. Regardless of fault-response architecture used, care must be taken to ensure that all potentially-suspect hardware is removed from use and control continues using functioning hardware or is transferred to redundant hardware.

Single Thread Response—Single thread responses are more appropriate for block-redundant architectures.

Advantages:
Single-response designs minimize the amount of design, test, and analysis needed to implement and verify the fault management design.

Disadvantages:
Disruptions to normal mission operations may occur. Ground recovery response may require mission outages to diagnose and possible return to the primary string. Response may involve power cycling of healthy hardware, which can induce additional failures.

Example: A response that reconfigures the entire attitude control block (sensor, sensor input, processor, actuator output, actuator) from the primary string to the redundant string.

Multi-Thread Responses—Multi-thread responses better accommodate architectures with some amount of unit and functional redundancy. Multi-thread responses use several different reconfiguration responses depending upon the set of suspect failed hardware.

Advantages:
Multi-thread response design provides a means to implement responses to minimize amount of hardware that is reconfigured. The amount of design, test, and analysis effort is greater than that of single-thread response, but less than that for multiple levels of responses.

Disadvantages:
Multi-thread response designs have disadvantages similar to single-thread design. Additional analysis and test is required to ensure that the appropriate response executes for the detected failures. Another disadvantage to multithread designs is the verification complexity to show the correct persistence of racing detection and/or responses was chosen.

Example: A response architecture that reconfigures power electronics for power faults, reconfigures attitude-control hardware for attitude faults, and reconfigures processor hardware for processor faults.

Multiple Levels of Responses—Multiple levels of responses are used to define response actions with different levels of reconfiguration or execution response-time needs. The responses can be tailored to sensor failures, actuator failures, power failures, data-transmission failures, processor failures, or anomalous behavior in which the failure cannot be clearly isolated.

The responses tend to vary in severity based upon severity of the effect of the faults. Figure 16 shows multiple levels of responses for different levels of failures.

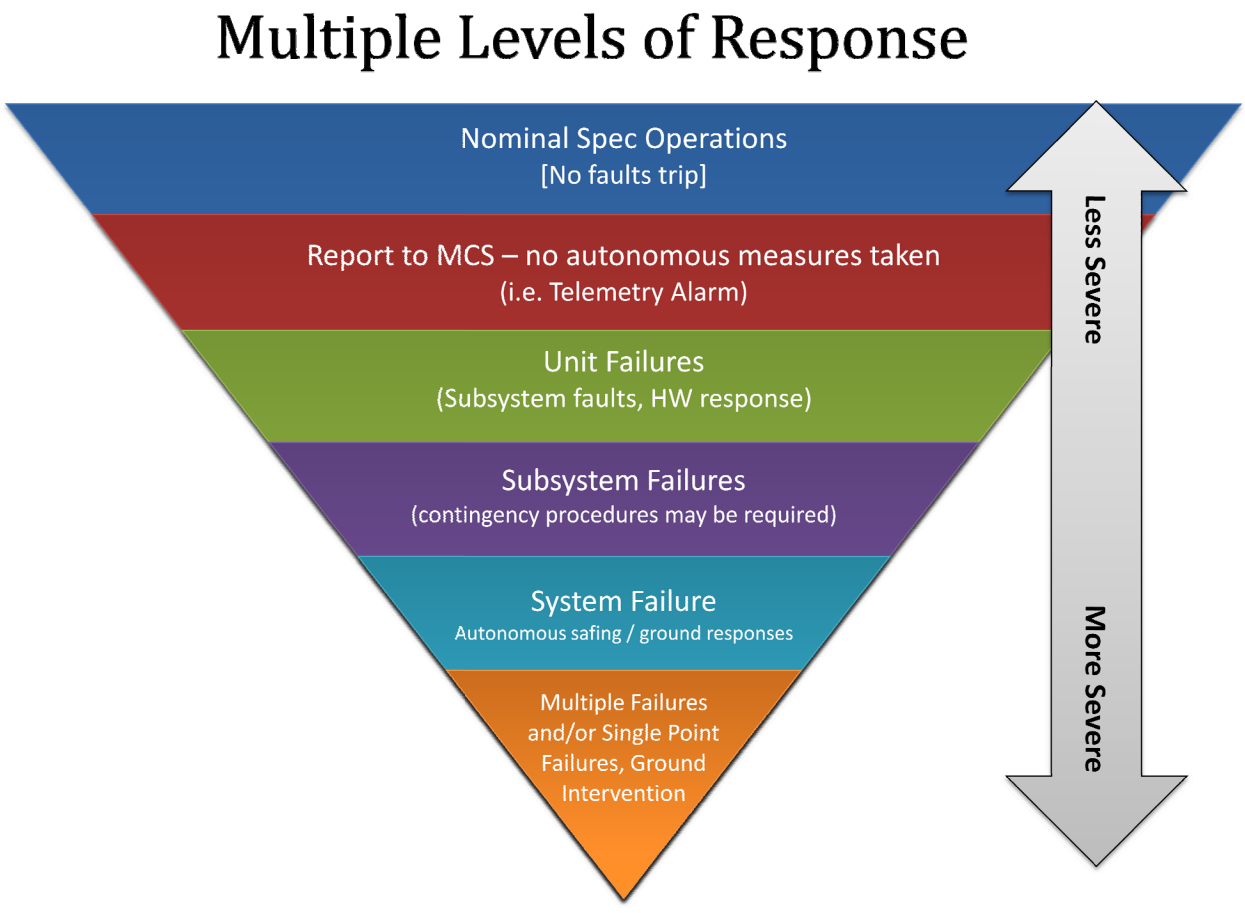


Figure 16. Fault Management response levels.

Advantages:
The use of multiple response levels minimizes disruptions to normal mission operations, in that either the software or hardware can reconfigure to no longer use suspect sensors and/or actuators with swapping to redundant processors. Ground recovery time is also reduced by minimizing the amount of hardware that is swapped to redundant units. Power swapping of healthy hardware can be minimized.

Disadvantages:
Multiple levels of responses increases fault management complexity. Careful identification of failure modes and failure observables is needed to design effective responses. Additional design effort is needed to define all of the responses, additional test and analysis effort is needed to ensure that the correct response executes for the observed failure and that the responses do not interfere with each other.

Example: One response is used to reconfigure the space vehicle data bus to the redundant side without swapping of other hardware units in response to a detected data bus failure. Another response would reconfigure to the redundant IRU for a detected IRU failure. Another response would reconfigure the attitude sensor, processor, and actuators for a detected attitude-control failure.

5.4.4.3.1 Fault Containment Techniques

The following types of response action may be taken: continued operation in a degraded mode, failure to a safe passive state; failure absorption; system reconfiguration; direct failure recovery; and operational limitation. In addition to taking response action, it is necessary to alert the operator to the failure. This is essential so the operator can modify control of the system as a result of the failure, or if the performance of the system is degraded. As stated previously, the methods used to alert the operator to the presence of failures are important, but will not be discussed in detail here [1].

5.4.4.3.1.1 Continued Operation in Degraded Mode

Telemetry to provide indications of performance degradation should be provided.

Rationale: Not all failures require response action. The failure may have no overall effect upon the system or may simply degrade system performance without significant effect on space vehicle safety. Alternatively, the failure may affect vehicle safety, but may be of sufficiently low probability that the effect can be accepted. It is, however, still necessary to detect and alert the operator to the fault so that maintenance action can be taken at the appropriate time. Loss of passive redundancy typically has no immediate effect on system operation. Subsequent failures may then have a severe effect, but no action is required as a result of the initial failure [1].

Example: Loss of one of several active star trackers will degrade attitude-determination accuracy, but will not have any immediate or significant effect on vehicle attitude control. No autonomous fault-response action is required. A deployment mechanism may jam, resulting in a failure to deploy and may have a significant effect on vehicle safety, but the probability of the failure may be sufficiently low that the effect can be accepted, without the need for autonomous fault-response action.

5.4.4.3.1.2 Failure to a Safe Passive State

A particular component may have hazardous failure modes but its continued operation may not be critical to the continued safe functioning of the system. In this case, on detecting the failure, the component may be switched to a safe passive state. In some cases the complete system may be shut down [1].

Rationale: Erroneous output from a thermistor or heater may be inhibited (e.g., by switching of relays) if there is active redundancy available to take-over the function. Thrusters could be shut down should an anomalous attitude rate occur during a thruster firing maneuver. Battery charge could be inhibited if the battery temperature becomes too high.

5.4.4.3.1.3 Failure Absorption

Failure absorption is achieved by nullifying the effect of the failure, normally by use of a voting process. This generally requires at least triplex redundancy so that the effect of the failure can be overcome by the action of the un-failed elements [1].

5.4.4.3.1.4 System Reconfiguration

If a failure occurs which degrades system performance below some acceptable level, then the system should reconfigure in order to recover an acceptable level of operation. Failure of an active redundant element will normally require changeover to a passive element [4].

Example: Responding to a processor failure by switching of control between units, or use of standby sensors or actuators. The system reconfiguration is not necessarily required to recover fully all aspects of system performance, but to maintain system performance at a level to maintain the vehicle in a power/thermal safe operating condition. System reconfiguration responses are by far the most common type of fault response used in autonomous space vehicle fault management.

5.4.4.3.1.5 Direct Failure Recovery

Certain types of transient failures may be directly recoverable, for example, failures caused by the natural space radiation environment, EMI, or by software errors. Failure recovery in these cases should be automatic or may require some action such as a processor reset. Typically, the number of resets is limited [1].

5.4.4.3.1.6 Operational Limitation

The final type of action is to place operational limits on the system. This may be achieved by restricting system functionality (effectively the same as degraded operation) or by providing instructions or warnings to the user or to other systems [1].

Examples: There may be an operational limitation (constraint) on the space vehicle orientation to the sun in order to protect sensitive instruments. The loss of a star tracker may result in the degradation of attitude determination performance. The loss of a torque rod may result in the degradation of the momentum-dumping capability and thrusters may be needed to dump momentum.

5.4.4.3.2 Space Vehicle Reconfiguration Fault Responses

Sensor Fault Responses—Sensor fault responses reconfigure either software to no longer use the suspect sensor data or hardware to power off the sensor and power on the redundant sensor. Because it is not always possible to isolate the failure to the sensor, sensor responses should swap to redundant sensor and input electronics if the fault persists. These responses usually need to execute before a more severe attitude control response triggers and must take into account any sensor warm-up time.

Actuator Fault Responses—Responses to command the faulty actuator to a safe state may not always be effective. In designing the response, it must be considered that the source of the failure lies in the ability to command the actuator. Removing power from the suspected failure actuator may be a more effective response. Analysis and test are required to ensure that the response does indeed result in a safe state for all of the possible set of faults. Actuator specific responses usually need to execute before a more severe attitude control response triggers.

Data-Transmission Fault Responses—Data-transmission fault responses can be a swap to an internal data bus or a full swap to redundant hardware. Data-transmission fault responses usually need to execute before a more severe attitude-control response triggers, or other functional failure that relies on data being transferred.

Power Fault Responses—Power fault responses are used to reconfigure equipment used for battery charge control and other power management functions. A low-power state can result in an under-voltage response.

Attitude Control Fault Responses—Attitude control fault responses are used to reconfigure sensors and/or actuators used for attitude. Often times these responses include transition to contingency modes or safe mode.

Thermal Control Fault Responses—Thermal control fault responses are used to disable thermistors or reconfigure heaters/ and/or heater control hardware.

Critical Event Responses—Fault responses may be designed to ensure the completion of critical events when required to establish a safe state. An example of a critical event would be a solar array deployment.

Processor Fault Responses—Processor fault responses are usually implemented in hardware. Processor fault responses need to have a fast response time, to allow control to be regained should the processor stop functioning or to minimize any damage resulting from a processor that operates improperly. Processor reconfiguration responses transfer processor operation from the primary processor to the redundant processer upon detection of a processor fault. The redundant processor can either be a cold spare (off) or a warm/hot backup (on with limited functionality or full functionality but not in control). In addition, processor responses need to include processor hardware resets to cover transient faults resulting from either software failures or radiation effects that cause processor failures.

Successful transfer to the redundant or backup processor is necessary to re-establish a functioning processor. A rigorous, systematic search, cross-strap fault management ECA, or sneak paths analysis is needed to prevent propagation of failures between the primary and redundant processor due to the processor failure detection and reconfiguration circuitry. The key to this investigation is a complete diagram of the involved interface circuits which penetrates each unit to a circuit depth sufficient to prove that no possible failures in one unit can propagate to become irreversible hardware failures in the second unit. Another input is a complete list of part- or assembly-failure modes which includes the following:

- Part failure modes (opens, shorts, stiction, etc.),
- PWB signal integrity,
- Single-event effects (latch-up, transfer, etc.), and
- EMI (latch-up, transfer, overvoltage).

These last three items are critical since they can affect both sides of a redundant pair [7].

Safety Net Responses—Fault responses may be designed to ensure overall non-specific space vehicle health. An example would be a fault response to space vehicle momentum out of tolerance, attitude errors beyond a set threshold, or attitude rates higher than the set limit.

5.4.4.4 Space Vehicle Fault Management Modes

There are many possible effective solutions based upon mission, orbit, space vehicle design, and mission availability (fly through mission) requirements. The suggested attributes of fault management contingency modes, safe mode, and the low-power response are provided in the following paragraphs.

5.4.4.4.1 Contingency Modes

Contingency modes are used to implement general purpose safing responses which are autonomously initiated by fault protection. Contingency mode safing sequences usually terminate mission operations, possibly suspend critical event sequences, and provide a general reconfiguration of space vehicle hardware to redundant or backup units. If necessary non-essential space vehicle loads are powered off to conserve power. Safing responses typically include the following general features:

> *Uplink communications*—The safing response and resulting contingency mode provides for a space vehicle state and attitude that ensures uplink for issuing space vehicle commands in the long term.
> *Downlink communications*—The safing response and resulting contingency mode provides for a space vehicle state and attitude that provides maximum telemetry coverage with positive link margins.
> *Environmental constraint*—The safing response and resulting contingency mode meets all boresight and radiator environmental exposure constraints.
> *Benign space vehicle configuration*—As a goal, safing responses and resulting contingency modes should place the space vehicle in an operationally benign state. This state may include, but is not limited to the following: (1) powering off all non-essential equipment including payloads when necessary, and (2) stopping of all non-essential space vehicle processes [7].

Commonly used contingency modes are earth-pointed modes, sun-pointed modes, or an inertial pointed mode. The selection of the contingency mode attitude is a function of orbit, mission, sensor suite, and the desire to quickly restore mission operations. In all cases a power-thermal, safe state that can be maintained for an extended period of time without ground intervention should be obtained.

5.4.4.4.2 Attitude Control Safe Modes

The attitude-control sensor and actuator suite are usually chosen to meet the attitude-control performance needs of the space vehicle. For many missions, a variety of attitude-control methods are needed with differing performance requirements. For example, following launch vehicle separation, thrusters and inertial reference units may be needed to damp large tip-off rates and sun sensors needed to quickly orientate the solar arrays to the sun to begin battery charging. Reaction wheels and magnetic torquers can be used as well for lower expected tip-off rates. Additional actuators and sensors are used for normal mission operations and depend upon mission and orbit. Communications space vehicles are usually earth-pointed with low agility requirements and tend to used earth sensors, star trackers, IRUs with thrusters, and momentum wheels. Vehicles requiring high-performance pointing accuracy tend to use star trackers and IRUs for attitude reference. Agile vehicles tend to use reaction wheels while highly-agile vehicles use CMGs, depending upon the level of agility required. Many vehicles use magnetometers and torque rods for momentum unloading and/or attitude control.

Post launch-vehicle separation attitude capture and vehicle activation control modes should be robust and these attitude-control modes have robust stability margins (gain and phase). The performance needs for normal mission operations may result in attitude control modes with relatively low gain margins (6 to 10 dB). In some cases, dynamic modes may even be phase stabilized.

The various attitude-control performance needs for post launch-vehicle separation attitude capture, space vehicle activation, normal mission operations, and de-orbit capability usually results in several different types of sensors and actuators. The more robust activation attitude control modes are commonly used for space vehicle safe modes. These modes are used to place the vehicle into a power/thermal safe operating condition with robust stability margins in event of an anomaly with the normal mission attitude control. Agile vehicles tend to have several safe modes, depending upon the severity of the fault. Descriptions of several safe mode architectures are provided in Aerospace document, TOR-2006(8606)-4494, section 10.1 [5].

5.4.4.4.3 Fully Independent Safe Mode

An ideal space vehicle safe mode would be fully independent and not re-use any components normally used for attitude control, including power supplies, processors, sensors, actuators, and communications paths. However, with the use of integrated avionics architectures it is more common to reconfigure only suspect equipment, sensors, actuators and control laws during safe-mode entries.

Advantages: Safe mode provides fault containment for a large variety of faults.

Disadvantages: Independent safe mode requires more hardware and increases the complexity of the fault management design. The additional hardware carries a weight and volume penalty. Also, safe-mode electronics are not modifiable after launch and hardware faults or design faults can render this mode unusable.

5.4.4.4.4 Low Power Response

Space vehicle designs should include a low-power response, which is designed to protect the space vehicle in the event of a short or a bus overload, or loss of attitude control. The software and/or hardware senses when the power drops below an established value for a specified time. If the criteria are met, all non-essential loads are shed from the bus [7]. The deepest level of low-power response should be implemented in hardware. If possible, critical space vehicle memories should be maintained throughout the under-voltage. Some space vehicles may need to reconfigure to a benign contingency mode following a transient under-voltage event.

Hardware implemented battery charge control will allow the batteries to be recharged whenever there is sufficient sun on the solar arrays and can prevent or extend the time to battery depletion. space vehicle designs exist that allow dead battery recovery and subsequent battery recharge, depending upon the nature of the fault that led to the under-voltage event, even after extended periods without power.

5.4.4.5 Fault Isolation

Fault isolation uses telemetry collected during the fault event and/or diagnostic procedures executed subsequently to determine as precisely as possible the specific cause of the fault. The space vehicle design (including the fault response) should store sufficient data during a fault to allow fault isolation. Where additional diagnostic information is available, the space vehicle design (including the fault response) should ensure that this is preserved for subsequent evaluation. The space vehicle design ideally provides the capability to exercise offline units in a non-intrusive manner, so that diagnostic tests can be performed.

5.4.4.6 Fault Recovery

Fault recovery uses the results of fault isolation to optimize the space vehicle configuration. When the fault response has taken the space vehicle out of service, a portion of the fault recovery activity is to bring the space vehicle payloads back online. Fault recovery may also involve additional space vehicle reconfiguration such as: restoring a primary unit switched out unnecessarily, switching out additional units not involved in the fault in order to simplify operations, and making adjustments to the fault management system. Fault recovery may also include updates to the ground system, including new alarm settings, new procedures, or even new operational concepts.

5.5 Fault Management Verification Techniques

The various fault management verification approaches, test environments, and analyses are described in the following paragraphs.

5.5.1 Fault Management Verification Approaches

5.5.1.1 Fault Branch Verification

Fault branch verification relies heavily on the data and structure provided by the fault branches of the fault tree analysis. Fault branch verification documents the verification evidence for the mitigation requirements for each branch of the fault tree. Verification methods are the standard, formal verification methods used to verify requirements, i.e. analysis, test, inspection, and demonstration. Figure 17 shows how the fault tree structure can be used to aid in the fault management verification planning and verification execution.

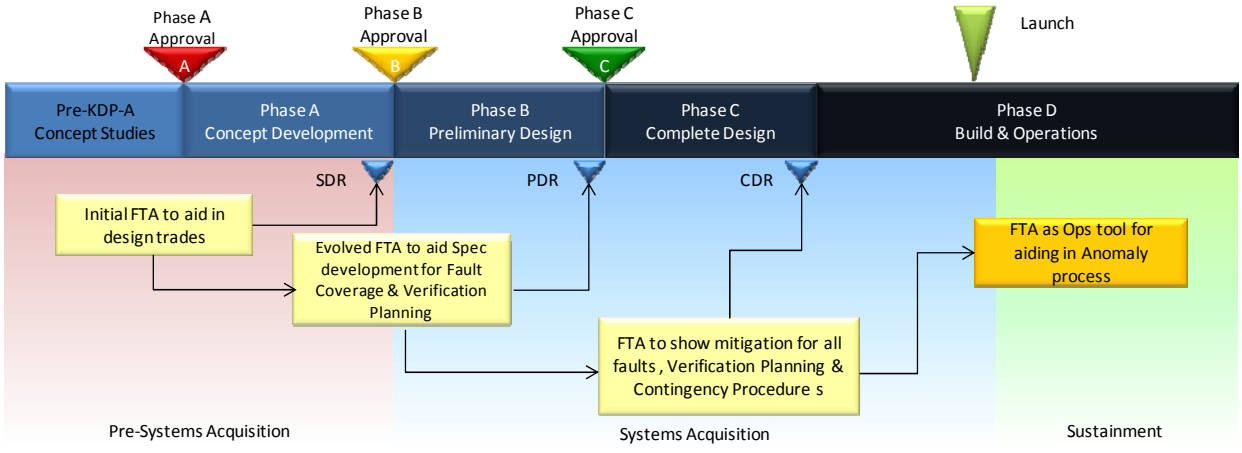


Figure 17. Fault tree analysis usage.

For example, if a fault branch is mitigated by design margin, then the verification should be an analysis that showed the required margin is met. If the fault branch is mitigated by any safety inhibits, then the verification should be the test and/or analysis showing that the inhibit performs as specified. If the fault branch is mitigated by an autonomous fault protection fault monitor and corresponding fault response, then the verification should consist of the following:

(1) The set of tests showing that the fault monitor was implemented correctly,

(2) The set of analyses and/or tests showing that the fault monitor correctly detects the fault (thresholds and persistence set correctly),
(3) The set of tests showing that the fault response was implemented correctly, and
(4) The set of tests and/or analyses showing that the fault response stops the fault from propagating and that any hardware reconfiguration or mode transitions put the space vehicle into a safe state.

Testing and/or analysis can be performed at the unit level, subsystem level, or system level. The tests can be a combination of formal verification demonstration tests, or informal "engineering tests." Regardless of type of test, formal or informal, the test results need to be documented and included as part of the verification evidence.

To support fault branch verification, the FTA should extend to the lowest level where fault detection is provided. Usually this is at the card, unit, or interface level. The FTA must also be kept up-to-date with all changes until formal verification is complete.

The advantages of fault branch verification is that it provides a framework for systematically documenting all of the various types of verification data needed to verify each of the fault mitigation requirements. Such a way to compile and document the verification data is needed when the distribution of the fault management requirements, and thus verification activities, span across multiple subsystems, multiple verification levels (unit, subsystem, system), and multiple verification methods, such as in integrated avionics architectures. It is not usually needed for simple fault management schemes which have few fault tests and fault responses and when the subsystem fault management requirements are decoupled.

5.5.1.1.1 Fault Branch Verification Matrix

The fault branch verification matrix (or table) is used to track and document the verification evidence for each fault tree branch. The fault branch verification matrix can be used early in the program (PDR) to identify what verification methods and verification levels will be used to verify the fault management requirements (hardware design, margin, and autonomous fault monitors/fault responses). As the verification planning matures, the fault branch verification matrix can be used to track the development of the verification analyses and test procedures. As the verification activities are completed, the fault branch verification matrix is populated with the references to the actual verification results.

Several metrics can be generated to track the fault management verification progress. For complex systems the fault management development and verification can span a few years. Metrics for reporting can be embedded in most common spreadsheet software. Useful metrics are: the number of verification procedures completed out of number required, the number of fault branches verified out of the number to be verified, etc.

Tables 5, 6, 7 and 8 show how fault branch verification matrices can be used at the various stages of verification planning, procedure development, and verification completion.

Table 5. Example: Fault Branch Verification VCRM

Fault Tree Branch	Fault Protection Method	Allocated Subsystem Requirement	Mitigation Rationale	Allocated Unit Level Specification	Inspection	FSW Verification	Space Vehicle I&T	Closed Loop Tests	Analysis
1.0 Loss of Attitude Control Examples									
1.1 Loss of Inertial Reference Unit	Autonomous fault management	Subsystem Requirement ID	Need to swap to redundant Inertial Reference Unit	Requirement ID	n/a	x	x	x	x
1.2 Loss of Star Tracker	Design margin	Subsystem Requirement ID	2/3 Active redundancy —continue on 2 trackers	Requirement ID	n/a	x	n/a	x	x
1.3 Excessive space vehicle Rates	Hardware implemented rate detection circuit and shutoff	Subsystem Requirement ID	Software errors can cause excessive rates, so protection provided in hardware	Requirement ID	x	n/a	n/a	x	x

Table 6. Fault Management Verification Plan

Fault Tree Branch	Fault Protection Method	Allocated Subsystem Requirement	Mitigation Rationale	Allocated Unit Level Specification	Inspection	FSW Verification	Space Vehicle I&T	Closed Loop Tests	Analysis
1.0 Loss of Attitude Control Examples									
1.1 Loss of Inertial Reference Unit	Autonomous fault management	Subsystem requirement ID	Need to swap to redundant inertial reference unit	Requirement ID	n/a	Test fault test logic, test fault response logic	Verify fault response swaps IRUs	Demonstration that fault response swaps IRU and continues to provide attitude control	Show that IRU swap works at high space vehicle rates
1.2 Loss of Star Tracker	Design Margin	Subsystem requirement ID	2/3 Active redundancy – continue on 2 trackers	Requirement ID	n/a	Verify logic to not use the failed star tracker	n/a	Verify response removes suspect star tracker and attitude control is maintained.	Show that attitude determination performance is met with any 2 of the 3 trackers
1.3 Excessive space vehicle Rates	Hardware implemented rate detection circuit and shutoff	Subsystem Requirement ID	Software errors can cause excessive rates, so protection provided in hardware	Requirement ID	Inspect unit verification data showing circuit performance is met	n/a	n/a	Verify detection thresholds and response	Show that detection threshold protects structural integrity and that it is not set too low causing fault triggers

Table 7. Example: Fault Management Verification Procedure Development

Fault Tree Branch	Fault Protection Method	Allocated Subsystem Requirement	Mitigation Rationale	Allocated Unit Level Specification	Inspection	FSW Verification	Space Vehicle I&T	Closed Loop Tests	Analysis
1.0 Loss of Attitude Control Examples									
1.1 Loss of Inertial Reference Unit	Autonomous Fault Management	Subsystem Requirement ID	Need to swap to redundant Inertial Reference Unit	Requirement ID	n/a	IRUTest1, IRUTest2	TESTXXX - IRU Swap Test Procedure	CLTEST XXX – IRU Failure	Memo ACS XXX
1.2 Loss of Star Tracker	Design Margin	Subsystem Requirement ID	2/3 Active redundancy – continue on 2 trackers	Requirement ID	n/a	STATest1, STATest2, STATest3	n/a	CLTEST XXX – STA Failure	Memo ACS XXX
1.3 Excessive space vehicle Rates	Hardware Implemented Rate Detection Circuit and Shutoff	Subsystem Requirement ID	Software errors can cause excessive rates, so protection provided in hardware	Requirement ID	Unit X VCRM	n/a	n/a	CL TEST XXX - Rates	Memo ACS XXX

Table 8. Example: Fault Management Verification Sell-off Matrix

Fault Tree Branch	Fault Protection Method	Allocated Subsystem Requirement	Mitigation Rationale	Allocated Unit Level Specification	Inspection	FSW Verification	Space Vehicle I&T	Closed Loop Tests	Analysis
1.0 Loss of Attitude Control Examples									
1.1 Loss of Inertial Reference Unit	Autonomous Fault Management	Subsystem Requirement ID	Need to swap to redundant Inertial Reference Unit	Requirement ID	n/a	IRUTest1 Results, IRUTest2 Results	TESTXXX - IRU Swap Test Procedure Results	CLTEST XXX – IRU Failure Restuls	Memo ACS XXX
1.2 Loss of Star Tracker	Design Margin	Subsystem Requirement ID	2/3 Active redundancy – continue on 2 trackers	Requirement ID	n/a	STATest1 Results, STATest2 Results, STATest3 Results	n/a	CLTEST XXX – STA Failure Results	Memo ACS XXX
1.3 Excessive space vehicle Rates	Hardware Implemented Rate Detection Circuit and Shutoff	Subsystem Requirement ID	Software errors can cause excessive rates, so protection provided in hardware	Requirement ID	Unit Data Package – section X.x.x shows circuit WCA and test results	n/a	n/a	CL TEST XXX – Rates Results	Memo ACS XXX

5.5.1.2 Failure Modes Effects Testing (FMET)

Patterned from an FMEA, fault management-effect testing induces failures into the hardware, software or system and verifies the correct system response and that the fault does not propagate. While in many cases conducting this testing is not practical, fault management-effects testing should be performed to the extent possible. For example, software responses to processor hardware faults can be verified during software unit testing when it is practical to inject or simulate hardware failures. Another example of FMET, is to induce failure signatures into closed-loop test beds and verify that the fault test(s) detected the fault and that the fault response succeeded in putting the vehicle into a safe state. For cases when failures cannot be physically induced or simulated, detailed analysis is needed to verify that the failures will produce the expected signature.

Regardless of fault management implementation methods used or level of complexity, FMET concepts should be used for all fault management testing.

5.5.1.3 Test-Like-You-Fly

Implementation of TLYF evaluation requires that the test environment provides a flight-like environment as possible. In many cases sticking to true TLYF goals in verifying fault management design requires conducting FMET.

5.5.1.4 Transition Matrix

A fault management mode transition matrix provides an effective method for capturing all of the possible mode transitions resulting from fault responses. All of these mode transitions should be verified by closed-loop test. This method works best for fault management systems that have multiple responses or several contingency modes. For multiple-level fault management architectures with parallel fault responses, this method can be overwhelmed by the number of possible transitions.

5.5.2 Effective Fault Management Testing

The overall goal to effective fault management verification is the integrated injected fault to response testing. It is important that the end-to-end verification be performed to ensure the fault protection behaves (detection, response, telemetry) in a predictable and correct way.

An effective fault management verification program is dependent upon complete testing of all of the fault protection related software and database values. Due to the various levels of implemented fault protection monitors and responses, a pyramid approach works best. The lowest level consists of all unit (hardware and software) level verification testing. This testing should include all testing of software exception handling and hardware error processing.

The next level of test verification consists of testing of all of the fault monitors and fault responses. For fault management architectures that have fault monitor and responses implemented across more than one unit, the verification of these fault monitors and responses should be performed in subsystem testing. ACS simulation test cases are needed to verify that the ACS fault-detection monitors and fault responses perform as expected under nominal and worst-case dynamic conditions. Closed-loop testing with flight software and attitude control dynamics is needed to both validate the ACS simulation results and to verify that the implemented fault monitors and responses perform as expected. Scenario-based testing should include end-to-end testing of the fault monitors

and fault responses in expected operational scenarios and/or on flight hardware. Scenario testing should include fault monitor and response testing for mission critical events.

Finally, fault management testing during space vehicle integration and test is used to verify the performance of fault monitors and responses that span across subsystems or hardware reconfiguration that cannot be fully verified during the lower level testing. Each level of testing should build upon the knowledge gained in the previous levels, allowing the number of test cases to be reduced. The complete set of tests should include testing of all database values, command sequences, and all logic paths. For some fault management implementations, there may be multiple fault monitors with a lesser number of fault responses. Each fault response should be tested end-to-end with at least one of the fault monitors triggering the fault response. Figure 18 shows the layers of fault management testing used to verify that requirements are satisfied

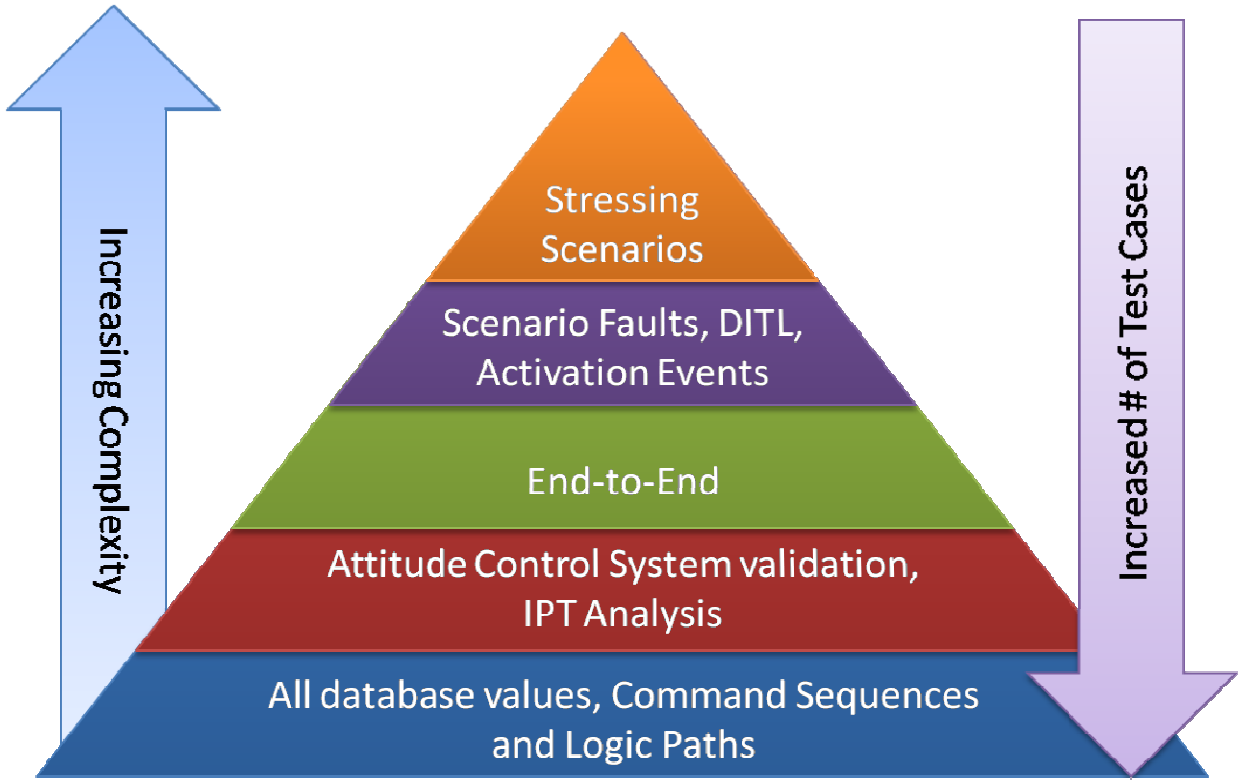


Figure 18. Fault management testing layers.

5.5.2.1 Hardware Unit Testing

The intent of fault management testing at the unit level is to ensure that the unit meets its part of the overall fault management requirements. The unit should be tested to show that the unit meets the fault management requirements allocated to it. To do this properly, the unit should be in a test environment as close to flight as possible. This includes interfaces with flight-like units, with flight-like timing, flight-like software and flight-like fault injection. There have been many examples of fault management testing that was not flight-like that resulted in a verified function but the simulated fault or event and real event were so different in signature that the fault management was defeated. It is also imperative that testing is done in all on-orbit modes.

Example: The Wide-Field Infrared Explorer Mission (WIRE) objective was to conduct a deep infrared, extra-galactic science survey. WIRE was launched on 4 March 1999, and was observed to be initially tumbling at a rate higher than expected during its initial pass over the Poker Flat, Alaska, ground station. After significant recovery efforts, WIRE was declared a loss on 8 March 1999.

Lesson(s) Learned

The WIRE Mishap Review Board has determined that the telescope instrument cover was ejected earlier than planned and at approximately the time the WIRE pyrotechnic electronics box was first powered on. The instrument's solid hydrogen cryogen supply started to sublimate faster than planned, causing the spacecraft to spin up to a rate of sixty revolutions per minute over the twelve hours following the opening of the secondary cryogen vent. Without any solid hydrogen remaining, the instrument could not perform its observations.

The root cause of the WIRE mission loss is a digital-logic-design error in the instrument pyrotechnic electronics box. The transient performance of components was not adequately considered in the box design. The failure was caused by two distinct mechanisms that, either singly or in concert, resulted in inadvertent pyrotechnic device firing during the initial pyrotechnic electronics box power-up. The control-logic design utilized a synchronous reset to force the logic into a safe state. However, the start-up time of the Vectron (Hudson, NH) crystal clock oscillator was not taken into consideration, leaving the circuit in a non-deterministic state for a time sufficient for pyrotechnic actuation. Likewise, the startup characteristics of the Actel™ A1020 FPGA (Mountain View, CA) were not considered. These devices are not guaranteed to follow their "truth table" until an internal charge pump "starts" the part. These uncontrolled outputs were not blocked from the pyrotechnic devices' driver circuitry. There has been no evidence or indication of any component failure although component failures were considered in the investigation.

A significant contributing cause of the anomaly was the failure to identify, understand, and correct the electronic design of the pyrotechnic electronics box. Design errors in the circuitry, which controlled pyrotechnic functions, were not identified. The pyrotechnic electronics box design was not peer reviewed, and other system reviews conducted by the instrument design organization did not focus on the electronics box. At the time the systems design review was conducted for WIRE the design of the pyrotechnic electronics box was not completed. It is the assessment of the WIRE Mishap Investigation Board that a peer review held during the design process, by people with knowledge of and expertise regarding pyrotechnic circuit design, would have identified the turn-on characteristics that led to failure.

A large number of failure scenarios were evaluated during the investigation to determine the cause of the cover ejection. These included; pre-launch, launch, powered flight, separation, software, operations, design, and component reliability faults. Based on comprehensive, systematic review of data, it was determined the cover was most likely ejected at the time the WIRE pyrotechnic electronics box was turned on, due to a transient condition that exists in the pyrotechnic electronics during startup. This transient condition is the direct result of the non-deterministic initialization of a field-programmable gate array (FPGA) that controls both the arming and firing circuits in the pyrotechnic electronics.

Although some design attention was given to the startup behavior of the FPGA, the design contained unidentified idiosyncrasies that triggered the cover ejection. The system design did not

contain sufficient start-up, lockout protection or independent provisions to prevent the FPGA startup operation from propagating to the firing circuits.

The anomalous characteristics of the pyrotechnic electronics unit were not detected during subsystem pyrotechnic or system-functional testing due to the limited fidelity and detection capabilities of the electrical ground-support equipment. Post-flight circuit analyses conducted as part of the failure investigation have predicted the existence of the anomaly, and it has been reproduced confidently using engineering model hardware [2].

Example: An interplanetary probe recovered from a major anomaly.

The spacecraft, designed to rendezvous with an asteroid, employed extensive autonomy because ground intervention during an emergency would take too long. The designers studied the history of an earlier project, which terminated prematurely after a data error depleted on-board fuel.

Three years into the flight, an engine burn aborted. A missing command in the burn-abort contingency command script prevented a graceful transition into the safe mode, and a series of anomalies ensued. Communication was lost for 27 hours before the flight computer regained control.

The initial script error was not caught during software tests. Hardware-in-the-loop simulation could not test abort scenarios because the brassboards were difficult to use. Exactly how the anomalies propagated is unclear because a bus undervoltage wiped out data from the recorder, nor could the anomalous behaviors be reproduced on ground.

During the emergency, the spacecraft fired its thrusters thousands of times. Fortunately, the fuel loss was tolerable because the thrusters were hard-wired to fire only for fractions of a second. The mission was saved because the designers took precaution against fuel depletion during a software crash, a lesson learned from the previous failure.

Lessons learned:

- Create extensive, realistic, nominal, and anomalous operational scenarios for testing at every level, from unit through system test.
- Implement robust simulators, including hardware-in-the-loop, for testing critical flight software functions.

Apply independent fault protection, such as hardware watchdogs, to mitigate risk in real-time systems, where errors can be so deeply buried as to be practically undetectable [6].

5.5.2.2 Software Unit Testing

Software fault tests and response testing should include all logic paths for the fault tests and responses, as well as any hardware error handling or exception handling code.

5.5.2.3 Subsystem Testing

The intent of fault management testing at the subsystem level is to ensure that the collection of units together meet the subsystem's part of the overall fault management requirements. The subsystem

should be tested to show that the subsystem meets the fault management requirements allocated to it, as there are usually complex unit and software interactions that are very difficult to simulate. To do this properly, the subsystem should be in a test environment as close to flight as possible. This includes flight-like units, with flight-like timing and traffic, flight-like software and flight-like fault injection. There have been many examples of fault management testing that was not flight-like that resulted in a verified function but the simulated fault or event and real event were so different in signature that the fault management was defeated. It is also imperative that testing is done in all on-orbit modes. The testing should also recognize the multiple dimensions involved in verifying that the design meets the intent of safeguarding the space vehicle. Single-thread testing (test that show a single thread or requirement has been met) may verify a single requirement by the letter of the wording of the requirement, yet miss the intent of the single requirement in the context of the overall intent. The multi-thread nature of a function (a collection of single thread requirements or functions) requires that the verification encompasses the group of requirements that together form a function in the context of how it will be used.

5.5.2.4 Closed-Loop Testing

Closed-loop testing consists of tests performed on test bed consisting of engineering model (EM) hardware and/or hardware simulations, flight code, and a dynamics simulation. It is also known as hardware in-the-loop testing (HITL). In some cases, the test bed can accommodate flight hardware as well.

For effective testing, it is important that this test environment be as flight-like as possible to ensure that the fault tests and responses are designed, specified, and implemented correctly, and to verify correct timing of signals and the responses. Where applicable to availability, EM hardware should be used in the simulation environment. High-fidelity EMs such as the flight computer and other control assemblies, should be incorporated to best simulate the actual flight hardware design. For components that are not practical to include, careful attention must be given to ensuring the simulated models and interfaces are as flight-like as possible. If not, then additional testing should be performed using actual flight hardware during system integration and test.

Early fault management verification planning is needed to procure and develop the closed-loop test environment. It is necessary to ensure that all needed EM hardware is either being developed as part of the program plan or is otherwise available.

Closed loop testing should be used to perform the fault test/response end-to-end tests, selected testing to validate ACS analysis and simulation results, testing to validate the interference, persistence and threshold analyses, and fault scenario tests to validate that the fault test and response actually put the space vehicle into the expected safe state for the injected fault.

Fault-scenario testing, introduces faults when the vehicle is in its expected operating condition, and verifies that the fault is detected by the expected fault test and that the fault response reconfigures the vehicle so that safe operation can continue. The scenario tests should simulate dynamic conditions needed to validate the ACS and IPT analyses. Fault scenario testing should include all tests to verify any “operate through” type of fault management requirements, in which a mission sequence must continue to transition the vehicle into a safe state. These conditions are typically any autonomous deployment sequences.

It is also possible to use the flight vehicle as the hardware in closed-loop testing if EM HW hardware is not available.

Example: Subject: In-flight spacecraft fault recovery.

Abstract:
Voyager 2 suffered two potentially mission-catastrophic, in-flight faults that were recoverable due to the availability of spare spacecraft hardware that was used to test workaround solutions. Maintain a functionally-identical, properly-configured, test bed of spare spacecraft hardware and associated support equipment, enabling detailed analysis of in-flight faults and candidate corrective actions.

Description of driving event:

During the prime Voyager mission, Voyager 2 suffered two potentially mission-catastrophic faults. These were: (1) failure of the spacecraft receiver plus significant degradation in the signal acquisition capability of the redundant spacecraft receiver, and (2) the azimuth scan platform actuator ceased operation. A significant contribution to the successful recovery from these faults was the availability of spare spacecraft hardware and associated support equipment, which was used to simulate postulated failures plus evaluate and validate proposed solutions and workaround procedures. The specific details of the corrections and workarounds are described in the paper entitled "Voyager Engineering Improvements for Uranus Encounter" by Howard P. Marderness [25].

Lesson(s) Learned:

The availability of spare spacecraft hardware and associated ground-support equipment can mean the difference between restoring a lost spacecraft capability or flying with a reduced or lost capability due to the inability to adequately correct a fault.

Recommendations:

Maintain a functionally-identical, properly-configured, test bed of spare spacecraft hardware and associated support equipment, enabling detailed analysis of suspected causes of in-flight faults and validation of proposed corrective actions or workarounds [6].

5.5.2.5 System Integration and Test

Hardware implemented fault management should be tested as part the functional test, performed before, during, and after environmental exposure. Software implemented fault management responses that reconfigure hardware should be verified in system-level test at least once. It is difficult to replicate all the flight harness, harness lengths, and units in the closed-loop test environment. Therefore it is necessary to verify that the proper hardware reconfiguration commands are sent, as well as proper timing of the responses on the flight vehicle in system test. Care must be taken to ensure that any fault management testing performed during system test, does not pose a danger to the flight hardware. Such precautions may limit the feasibility of some tests.

5.5.3 Fault Management Verification Analysis

5.5.3.1 Attitude Control Simulations

Simulation and modeling has proved a valuable aid to system developers in refining monitoring algorithms and other system-functional requirements. The use of simulation allows the effectiveness and robustness of monitors to be tested more thoroughly than is possible with the real

system and allows testing and validation in advance of the implementation [1]. Simulation is particularly useful in developing and verifying ACS monitor algorithms, fault tests, responses, thresholds, persistence and potential interference.

ACS simulation cases should be used to verify that fault tests and responses perform as expected for off-nominal dynamic conditions, sensor failures, and actuator failures. The various vehicle operating and failure induced dynamic conditions should be included. These simulations should also include all contingency mode entries for the range of failure induced dynamic conditions.

5.5.3.2 Threshold, Persistence, and Interference Analysis

Fault monitors should include provisions for minimizing the likelihood of spurious fault indications —provisions termed spurious signal negation. These can include filtering, persistency, or cross-checking. Fault monitors should also include provisions to resolve ambiguous or common-mode failures pyrotechnic those which could trip multiple fault monitors. Ideally, the space vehicle design will allow the fault monitors and responses to be independent. When this is not possible, the fault monitors need to be prioritized, or the associated fault responses need to include some embedded response logic to avoid having the fault responses interfere. The analysis to determine that the fault monitors have the proper persistence, thresholds, and that the fault responses do not adversely interfere with each other is referred to as either IPT (Interference, Persistence, and Thresholds) or TPI (Threshold, Persistence, and Interference) analysis.

A TPI analysis should always be performed as part of the fault management verification activities. The extent of the analysis will depend upon (1) the number of fault monitors that use threshold to detect a failure, (2) the number of fault monitors with persistence counters (should be most of them), and (3) the number of faults monitors that cannot be shown to be independent. Attitude control fault monitors and responses typically tend to have the potential for interference, as it is difficult to distinguish between actuator and sensor faults. The problem is exacerbated when multiple actuators may be in used, such as momentum wheels, CMGs and thrusters. Interference may also occur when fault responses autonomously reconfigure hardware. It is also important to ensure that the responses are designed such that another subsequent response does not reconfigure back to suspected failed hardware.

The TPI analysis used for fault management verification should contain the information described as follows. Program-unique interaction or considerations that are used in the TPI analysis should be described in the analysis portion or in the description summary for each TPI analysis entry. These descriptions serve two purposes. The first is to understand the rationale behind the selected threshold and persistence values. The second purpose is to provide guidance for future iterations of the analysis that will occur throughout the life of the space vehicle as components or subsystems degrade or fail. Often the TPI analysis will refer to the fault tree and branch termination analysis.

The TPI analysis provides a summary of each fault monitor and the rationale for the threshold and persistency values. For Boolean threshold values, the value used should indicate that the test should fail. For example, if a test should fail when the value goes to “1”, the value of the threshold should be “1.” The “predefined values” are referred to as “thresholds” or “trigger points,” and represent the value at which an anomalous condition is present. A fault monitor may also include logic which detects for, and ignores, failed sensors. “Consecutive occurrence counters” are used in some space vehicle; these are referred to as “persistence filters” and may be used for a variety of reasons: to ensure transient occurrences do not trigger a response, to satisfy hardware turn-on constraints, or to allow other fault-protection algorithms to detect faults first. Persistency can be thought of as the duration of a fault condition.

Threshold values and persistence values should always be supported by analysis and ACS simulation cases for ACS faults. These analyses and simulations should show that the chosen thresholds and persistence values actually protect the hardware by not allowing a damaging condition to persist to the point of causing damage or unrecoverable event. They also must be chosen to prevent false alarms. The threshold and persistence values should also be verified to be correctly implemented by test. ACS simulations are also needed to verify that the ACS fault tests and responses do not adversely interfere with each other. The dynamic, interference-simulations results should also be verified by closed-loop test. Usually only a subset of the most stressing simulation cases are used for the closed-loop testing.

The interference part of the analysis should systematically show that the fault monitor and responses do not negatively interact with each other. The interference analysis can show that the faults and corresponding responses are independent of each other, or that the thresholds and persistence limits are set to ensure that the correct response is activated. In addition, logic may need to be used to ensure that a potentially interfering response is not inadvertently triggered by the effects of the fault.

The TPI analysis is considered a “living document.” It is expected that it will be periodically updated throughout testing and the life of the space vehicle. There are usually certain threshold values that need to be updated as better knowledge of the space vehicle’s hardware, dynamics, and flight software behaviors are understood.

5.5.4 Fault Coverage Analyses

System sell-off of fault management should include analyses showing the fault coverage and that the fault monitors, responses, and fault management modes result in fault containment allowing operations to continue or safe transfer to a degraded operating mode.

Just as the FTA and FMEA can be used to identify fault-detection requirements, they can also be used to show the fault coverage provided by the various fault-mitigation and protection techniques used.

Analysis methods for assessing fault coverage and the effectiveness of the fault management to contain faults are described below.

5.5.4.1 Branch Termination Analysis

Fault tree branch termination analysis shows that the fault requirement set provides protection for the fault tree branches and ultimately for the identified root conditions.

The fault branch termination analysis shows how built-in safety devices, design margin, constraint checking, on-board fault protection, or ground-based recovery procedures avoid functional losses and root conditions to maintain on-orbit mission operations. Fault branch termination analysis results can be presented in several ways and the use of a matrix or table form are very effective. For many branches, the analysis shows how the total space vehicle design avoids its lower branches by over-design, redundancy, or procedures. Some branches do not have direct fault protection; they are protected if all branches above them or below them are protected. Protecting all branches above ensures the fault below cannot be reached. Protecting all branches below insures that each cause of a fault can be isolated so that the fault can also be contained. A complete fault design usually has every branch below every root condition terminated (covered).

The branch termination analysis demonstrates how the fault design maintains functionality though operator errors, hardware failures, and space environment upsets. The fault tree also shows the possibilities that the fault design must tolerate. The branch termination analysis a specific part of the fault design that accounts for different possibilities. A short description between every end branch and the top of the fault tree show how the end causes are accounted for.

Each entry of the branch termination analysis provides unique data about possible faults. Each entry also contains the FT identifier and description of each fault or anomaly. A mission mode identifier is can be provided because the space vehicle may change configurations and may have different protection schemes. The effect description describes how the fault causes problems. The detection describes the method for detecting that a fault occurred, either by an autonomous fault monitor or by the ground. The corrective action or response describes the on-orbit or ground response that is needed to isolate the fault. Each branch may identify faults that either requires a ground response or space vehicle autonomous response.

Just as the FTA and FMEA can be used to identify fault detection requirements, they can also be used to identify requirements for fault recovery. It is, however, an iterative process because the requirements for fault recovery may change the system design which in turn will modify the analyses [1].

5.5.4.2 FMECA Coverage

The FMEA provides all of the identified hardware failure mode effects. To show that the fault management design provides protection for the identified failure modes, an analysis is needed to show that each FMEA item that requires an autonomous transition to redundant hardware or a ground response will be detected by the fault monitors.

5.5.4.3 Single Event Effects Coverage

If single-event effects resulting from off-nominal conditions, such as an anomalously large solar flare, need to be protected against by fault management, an audit similar to the FMEA audit is needed to ensure that any single-event effects are corrected to allow either mission operations to continue or to transition to a fault management mode.

5.5.4.4 Mission Critical Events Analysis

Mission-critical events in the operation of a space vehicle are those which, if not executed successfully (or recovered from quickly in the event of a problem), can lead to loss or significant degradation of mission. Included in critical event planning are timelines allowing for problem identification, generation of recovery commands, and up linking in a timely manner to minimize risk to the in-space assets. Examples include separation from a launch vehicle, critical propulsion events, deployment of appendages necessary for communication or power generation, stabilization into a controlled power positive attitude, and entry-descent and landing sequences.

6. Definitions and Common Terminology

Active reaction—A reaction to a fault that includes some manner of space vehicle reconfiguration to accommodate the fault, typically by exchanging offline and online resources.

Algorithm—A precise rule (or set of rules) used to solve anomalies.

Anomaly —Any deviation from expected performance associated with remotely operated elements(s). Anomalies (and faults) can include hardware failures, recoverable hardware upsets, infrequent extreme environmental conditions, or operator errors.

Anomaly Detection and Resolution (ADR)—The features and processes associated with the handling of anomalies. Synonymous with *Fault, Detection, Isolation and Recovery (FDIR)* in the NASA lexicon.

Autonomy—Ability of the remotely operated element (ROE) to detect, diagnose and respond to anomalies and other conditions without Operations Center action.

Blind Operations (BlindOPS) —Operation of a remotely operated element in absence of diagnostic information.

Category I (CatI)—Designation given to anomalies that can cause catastrophic or permanent loss of mission if unabated. Characteristically, these invoke protective measures. See *Level 1 Fault*

Category II (CatII)—Designation given to anomalies that can cause loss of mission if unabated.

Category III (CatIII)—Designation given in anticipation of all other anomalies that may occur, but do not have pre-established ADR. These are typically resolved at a technical support center (TSC).

Common Cause Failure—Similar components, specifically those of identical design manufacture, can fail as groups where they share a common defect are put into a common, redundant environment. Failures can stem from common manufacturing defects in similar components, failures induced by incorrect maintenance actions taken while servicing a set of redundant components, and failures stemming from environmental extremes such as excess temperature, excess mechanical stresses, and normal of failure-induced electromagnetic interference [19].

Common Mode Failure—See *Comman Cause Failure*.

Containment Region (CR)—The levels at which the anomaly(ies) effect(s) are experienced (e.g. component, subsystem, etc.)

Coverage—The design reference set of anomalies covered by ADR (e.g. Category I and II Anomalies).

Cyclic Redundancy Check (CRC)—A cyclic redundancy check is a type of function that takes as input a data stream of any length, and produces as output a value of a certain space, commonly a 32-bit integer. The term CRC denotes either the function or the function's output. A CRC can be used as a checksum to detect accidental alteration of data during transmission or storage.

***Database*—**General reference to those stored values that support ADR (e.g. look-up tables, calibration curves, stacked command sets etc)

***Design Assurance*—**The traceable systematic multi-level activity ensuring accurate translation of all requirements, specifications, and standards into a detailed producible, testable, supportable design.

***Design Diversity*—**Use of components and software of different designs to tolerate design failures [20].

***Design Reference Anomalies (DRA)*—**See coverage.

Detection (D) –Same as *Fault Detection.*

***Detection Verification (DV)*—**Features and processes associated with ensuring that the detected anomaly(ies) are correct. This is usually implemented with features associated with spurious signal negation.

***Downing Event (DE)*—**Any anomaly that results in the removal of the remotely operated assets from operational service. These events are counted against the "uptime" fraction (numerator) of both inherent and operational availability requirements.

***Element*—**The general level of designation for the asset being control (e.g. launch vehicles, satellite vehicle, operations center are at the element level designation).

***Fail-Operate*—**In the event of failure the system will continue to operate and remain in a safe, possibly degraded, state [16].

***Fail-Safe*—**In the event of failure the system will revert to a mission non-operating state that will not cause a further mishap [16].

***Failure*—**An unexpected response whereby the function is not recoverable.

***Failure Mode*—**The way in which a component fails [19].

Failure Mode Effects Analysis (FMEA)—A procedure by which each potential failure mode in a system is analyzed to determine the results or effects thereof on the system and to classify each potential failure mode according to its severity [11].

***False Alarm Negation*—**See spurious signal negation.

***Fault*—**An unexpected response whereby the function is recoverable, either by fixing it, managing around it, or by redundancy.

***Fault Detection*—**Features and processes associated with perceiving that a failing to perform has occurred. AKA Fault Monitor(ing).

***Fault Coverage*—**The probability that the system can recover, given that a fault occurs [19].

***Fault Isolation*—** *(i)* Features and processes associated with diagnosing where the anomaly(ies) occurred. *(ii)* The analysis to determine what component has failed, how it has failed, and why it

failed. Fault isolation is done by analyzing the satellite data collected either during the period immediately preceding the failure, or during tests specifically performed to investigate the causes of the failure. Sufficient data must be available for successful fault isolation.

Fault Identification—The process of isolating a fault in a system to a specific function.

Fault Management—See Fault Management Systems Engineering.

Fault Management Systems Engineering—An engineering discipline that address the occurrence of faults on space vehicles and provides a means for reducing their effect through cooperative design of both the space vehicle and ground elements and operator actions. In safety terms, this would be safety engineering.

Fault Mitigation—The passive mitigation of faults through design. Examples are, ARM/EXECUTE command pairs for hazardous commands, active redundancy, extra solar array strings, etc. Knowledge of the fault occurrence is not necessary to mitigate the effect of the fault. In safety terms, this is the step to either eliminate or mitigate the hazard (aka fault).

Fault Monitors—The portion of Fault Protection that detects faults.

Fault Protection—The implementation of design intended to actively control the effects of a fault.

Fault Recovery - The action taken to bring a system back into operation after a fault has occurred. Fault recovery (e.g., switching to redundant components, resetting a processor, etc.) obviously depends on the availability of satellite resources.

Fault Responses—The portion of Fault Protection which reconfigures the space vehicle to contain the fault.

Fault Tolerance—The number of faults that the system must tolerate to meet its specifications. That is, a single fault tolerant space vehicle must still be able to meet its mission requirements after a single fault.

Fault Tree Analysis (FTA) —Fault Tree Analysis is a deductive, failure-based approach. As a deductive approach, FTA starts with an undesired event, such as failure of a main engine, and then determines (deduces) its causes using a systematic, backward-stepping process. In determining the causes, a fault tree (FT) is constructed as a logical illustration of the events and their relationships that are necessary and sufficient to result in the undesired event, or top event [11].

Flag—An alert generated when a specified value has been exceeded (e.g. Red flag indicates that temperature dropped below survival limit).

Functional Failure Analysis (FFA)—The capture document that integrates data related to design reference anomalies.

Graceful Degradation—The property in a system that allows a system to continue operating properly in the event of the failure of some of its components. Generally implemented at the lowest level of cross-strapped functions.

***Hardware Faults*—**Inherent defects that are found in any manufactured hardware item. (Dunn, 2002).

***Hardwired Telemetry*—**The most basic set of Telemetry available without an operational flight computer allowing visibility to critical system status and low level functions.

***Hazard*—**(i) Any real or potential condition that can cause the damage or loss of a system [18]. (ii) any real or potential condition that can cause damage to or loss of a system, equipment, or property; or damage to the environment.

***Hierarchy*—**Diagnostic and response order of precedence following the detection of an anomaly. This typically follows the hierarchy of the system indenture.

***Human-in-the-Loop (HIL)*—**Presence of a human operator in ADR. Other variants include operator-in the-loop, operator-assisted and human-assisted designations.

***Incipient Failure*—**General reference to a characteristic trend that indicates an item will fail, but has not yet reached its out-of-specification limit.

***Independent Review Team*—**A team encompassing the Mission Assurance Team (MAT).

Independent Readiness Review Team (IRRT), and other independent review activity teams [1].

Inhibit (INH)-Logic features and processes that precludes an action from occurring in advance (e.g. locking out a degraded sensor before it generates a false signal). Same as Lock-out and opposite of Enable. Alternately referred to as Disable.

***Intrusive Check-out (ICO)*—**Diagnostic relying on operationally disruptive means in order to diagnose the anomaly (e.g. powering on and off cross-strapped redundant strings of a communication system to isolate the failed item.)

***Isolation*—**Same as *Fault Isolation.*

***Lock-out (LO)*—**Disabling or ignoring a specific measurand. Example lock-out of a bad sensor in a voting logic circuit.

***Majority Vote*—**Logic that defines a prevailing action (e.g. any two of three independent signals exceeding a limit check is cause for the initiation of an alarm).

***Masking*—**Features and processes that preclude an anomaly from becoming a downing event (i.e. anomaly occurs, but has no effect on the mission).

***Maximum Time to Mission Loss (MaxTTL)*—**The upper limit, usually specified at some statistical confidence bound, to respond before a downing event is experienced.

***Maximum Time to Restore (MaxTTRe)*—**The upper limit, usually specified at some statistical confidence bound, to restore the remotely operated element following a downing event.

***Mean Time to Restore (MTTRe)*—**The average time required to restore the remotely operated element following a downing event.

Measurand—A specific measurement or parameter (e.g. voltage measurement on the battery terminals).

Mishap—Unplanned event or series of events resulting in damage or loss of mission [18].

Mishap Risk—An expression of the impact and possibility of a mishap in terms of potential mishap severity and probability of occurrence [18].

Mishap Severity—Defined as four categories: catastrophic, critical, marginal, and negligible [18].

Mission Critical Failure—A fault resulting in significant or total loss of the ability to meet any minimum mission performance specifications prior to reaching the end of design life.

Mission Degrading Failure—A fault resulting in partial loss of the ability to meet any minimum mission performance specifications, or the total loss of one or more system capabilities, prior to reaching the end of design life.

OBFault management—On Board Fault Management.

Operations Center (OC)—Assets where command and control authority for the remotely operated element (e.g. mission control, space vehicle operations center, etc.).

Override (OVR)—Logic features and processes that allow reversal or negation of an action being taken after the action was taken (e.g. return to operation after autonomously entering safemode).

Passive reaction—A reaction to a fault that takes advantage of pre-existing online redundancy.

Protective Measures (PM)—Features and processes that are invoked when a Cat I anomaly occurs (e.g. emergency power load shed). See *Fault Management*.

Random Hardware Failure—Defect or failure whose occurrence is unpredictable in absolute sense, but is predictable in a probabilistic or statistical sense.

Recovery (Re)—Synonymous with restoration and/or return to service following an anomaly(ies). Also see *Fault Recovery*.

Remotely Operated Element (ROE)—The asset subject to ADR requirements (e.g. satellite, launch vehicle, missile, unmanned aerial vehicle, etc.).

Resolution (R)—Features and processes associated with responding to the anomaly(ies). Also see *Fault Responses*.

Resolution Verification (RV)—Features and processes associated with ensuring that the expected response taken against the anomaly(ies) matches that achieved.

Risk—Refers to the events that are possible, but not yet realized, and that carry adverse consequences for a program or mission. Risk is usually characterized by the identification of the risk events that pertain to a specific program or mission (by their probability of occurrence), and by the magnitude of the possible impacts as measured in some appropriate scale of assessable consequences.

Safe Hold Mode—*(i)* A reduced functional fail-safe Space Vehicle mode in which a balanced power-safe, thermal safe, and communications-safe state is maintained allowing ample time for Ground Operations to recover to an operational Space Vehicle mode. *(ii)*A specific end state, invoked by protective measures, in which the asset(s) at risk due to an anomaly(ies) is placed in a safe condition. *(iii)* A mode developed on the space vehicle to safe the vehicle from different types of failures. There may be one or more modes based on severity of the failure. See *Contingency Mode.*

Safing—Process of preserving the assets at risk due to the anomaly(ies).

Software Faults—Inherent defects that can reside in the software as a result of software programming[18]. Software Faults are also considered defects that reside in the software as a result of software requirements. Software faults occur when software does not produce a correct response given a set of inputs and internal states. They may stem from personnel errors (logic or typos) or process failures in code generation while being integrated on the ground.

Space Vehicle—a craft capable of traveling in outer space; technically, a man-made craft in orbit around the earth.

Space Vehicle Anomaly—Unplanned event or series of events potentially resulting in damage or loss of mission.

Space Vehicle Hazard—Any real or potential condition that can cause loss or damage to a launched mission. (includes causes of hardware failures or external events).

Space Vehicle Mishap—Unplanned event or series of events resulting in damage or loss of mission.

Space Vehicle Safety—The coordinating activity to reduce the risk of loss of mission to launched space vehicles.

Spurious Signal Negation—Features and processes that preclude an anomalous signal from causing an incorrect action (e.g. false alarm, premature reconfiguration, etc.).

Stored State of Health (SSOH)—Onboard storage of telemetry available after durations when out of view or leading up to an anomalous condition.

Survival Mode—A fail-safe space vehicle mode in which a balanced power and thermal state is obtained to maintain space vehicle in a recoverable state (may be passive or active control).

Symptom—Characteristic sign or indication of the existence of an anomaly.

System Safety—The coordinating activity taken to guarantee that the final product under development will be safe [19].

Systematic Faults—Category of faults that includes Personnel error, Environmental conditions, Design inadequacies and Procedural deficiencies. Includes all faults beyond hardware and software faults that can be introduced anywhere in the system life cycle including the design, implementation, operation, and maintenance phases [19]. See *Common Mode Failures*.

Technical Support Center (TSC)—Assets where engineering and scientific support for the remotely operated element exists (e.g. auxiliary support center, engineering service facility, customer support center, etc.). Typical functions include: coordinating requirements for ROE database updates and delivery, provide technical advisor support, collect and interpret of ROE data, and provide a center for management support of the mission.

Telemetry (TLM)—Transmission of data from the remotely operated element to the operations center.

Thread—A sequence of events that defines the ADR schema.

Time to Critical Effect (TTCE)—The bounding case (i.e. minimum timeline duration) at which the anomaly(ies) manifest effect(s) if unabated.

Time to Irreversibility (TTI)—The bounding case (i.e. minimum timeline duration) for which the anomaly(ies) must be responded to. Note: TTI must always be a shorter duration than the Time to Critical Effect.

Timeline Decomposition—The process by which operational threads are evaluated against critical event threads.

Time-out (TO)—A delay introduced for the purpose of allowing a human operator action to intervene.

Transition Matrix—A method of capturing the state and mode changes between critical events. This is often used to provide an exhaustive check on conditions for autonomous operations and rules defining operator enabled actions.

Validation—Methods used to show specification are appropriate.

Verification—methods used to show specifications are met.

7. Fault Management Guidelines by Space Vehicle Class

Table 9. Fault Management Functional Guidelines

Fault Management Mode Guidelines 4.1.3.3	Class A	Class B	Class C	Class D
Contingency Modes	Recommended	Recommended	Optional	Optional
Contingency Modes - Positive power/thermal balance	Recommended - recommended to provide positive power/thermal balance if used	Optional - recommended to provide positive/thermal balance if used	Optional	Optional
Contingency Modes - Uplink Communications	Recommended - recommended to provide uplink commandability with positive link margins if used	Optional - recommended to provide uplink commandability with positive link margins if used	Optional	Optional
Contingency Modes - Downlink Communications	Recommended - recommended to provide telemetry with positive link margins if used	Optional - recommended to provide telemetry with positive link margins if used	Optional	Optional
Contingency Modes - Mission Operations	Recommended - Limited mission operations, recommended to protect against Mission Payload environmental constraint violations if used	Optional - Limited mission operations, recommended to protect against Mission Payload environmental constraint violations if used	Optional	Optional
Safe-Mode				
Safe-Mode - Positive power/thermal balance	Recommended to provide positive power/thermal balance if used	Recommended to provide positive/thermal balance if used	Optional	Optional
Safe-Mode - Mission Operations	Recommended - 1) power off all non-essential equipment, 2) stop all non-essential activities	Recommended - 1) power off all non-essential equipment, 2) stop all non-essential activities - NASA PD-ED-1243	Optional	Optional
Safe-Mode - Attitude Control	Recommended	Recommended	Optional	Optional
Safe-Mode - Electrically Independent Processors	Recommended	Recommended	Optional	N/A
Safe-Mode - Uplink Communications	Recommended to provide uplink commandability with positive link margins	Recommended to provide uplink commandability with positive link margins - NASA PD-ED-1243	Optional	Optional

Safe-Mode - Downlink Communications	Recommended to provide telemetry per Telemetry coverage requirements with positive link margins	Recommended to provide continuous telemetry coverage with positive link margins - NASA PD-ED-1243	Optional	Optional
Safe-Mode Duration	Recommended - 60 days LEO, 90 days HEO/GEO	Recommended Two Weeks - NASA PD-ED-1243	Optional	Optional
Low Power Response				
Low Power Response - Deterministic State	Recommended to provide deterministic spacecraft state	Recommended to provide deterministic spacecraft state	Recommended to provide deterministic spacecraft state	Recommended to provide deterministic spacecraft state
Low Power Response - Load Shed	Recommended - Independent (Hardware) load shed of all non-essential Loads	Recommended - Independent (Hardware) load Shed all non-essential Loads - NASA PD-ED-1243	Optional	Optional
Low Power Response - Charge Control	Recommended - Independent (Hardware) Battery Charge Control	Recommended - Independent (Hardware) Battery Charge Control	Optional	Optional
Low Power Response - Critical Telemetry	Recommended - Independent (Hardware) Critical Telemetry	Recommended - Independent (Hardware) Critical Telemetry	Optional	Optional
Low Power Response - Command Capability	Recommended - Independent (Hardware) Command Capability	Recommended - Independent (Hardware) Command Capability	Optional	Optional
Low Power Response - Survival Heaters	Recommended - Ground control of thermo-static Survival heater enable/disable state	Recommended - Ground control of thermo-static Survival heater enable/disable state	Optional	Optional
Dead Bus Recovery Capability	Recommended	Recommended	Recommended	Recommended unless short term environmental conditions would preclude ability to continue mission
Fully Independent Safe Mode	Optional	Optional	Optional	N/A
Manual Recovery	Recommended	Recommended	Optional	Optional
Fault Management Mode Recovery	Recommended	Recommended	Optional	Optional
System Disposal	Recommended	Recommended	Optional	Optional

System Disposal Fault Tolerance	Recommended	Recommended	Optional	Optional
Independent System Disposal	Recommended if uncontrolled re-entry poses human safety hazard or security breach	Recommended if uncontrolled re-entry poses human safety hazard or security breach	Recommended - Independent Disposal capability (De-orbit, other)	Not recommended
Independent System Disposal Fault Tolerance	Recommended if uncontrolled re-entry poses human safety hazard or security breach	Recommended if uncontrolled re-entry poses human safety hazard or security breach	Recommended - Independent Disposal capability (De-orbit, other)	Not recommended

Table 10. Fault Management Requirements Derivation Guidelines

Fault management Requirements Derivation 4.1.3.4	**Class A**	**Class B**	**Class C**	**Class D**
Space Vehicle vs Ground	Recommended	Optional	Optional	Optional
Protection Against Operator Errors	Recommended	Optional	Optional	Optional
Protection Against Space Vehicle Hardware and Software Design Errors	Recommended	Optional	Optional	Optional

Table 11. Fault Management Performance Guidelines

Fault Management Performance Guidelines 4.1.3.4.3	**Class A**	**Class B**	**Class C**	**Class D**
Single-Fault Tolerance	Recommended for all activation and Mission modes	Recommended for all activation and Mission modes	Optional	Optional
Enhanced Single-Fault Tolerance	Recommended for all activation and Mission modes	Recommended for all activation and Mission modes	Optional	Optional
Multiple Fault Tolerance	Recommended	Recommended - NASA PD-ED-1243	Optional	Optional
Fault Coverage	Recommended for random hardware faults, Hazardous vehicle constraints, hazardous vehicle conditions, off-nominal SEU effects	Recommended for random hardware faults, Hazardous vehicle constraints, hazardous vehicle conditions, off-nominal SEU effects	Optional	Optional
Safe Mode Entry & Exit Fault Tolerance	Recommended no Single failure prevent successful entry or exit	Recommended no Single failure prevent successful entry or exit	Optional	Optional
Independent Command & Telemetry	Recommended for all activation and Mission modes	Recommended for all activation and Mission modes	Optional	Optional

Table 12. Fault Mitigation Design Guidelines

Fault Mitigation Design Guidelines 4.1.3.4.4	Class A	Class B	Class C	Class D
Fused Power Considerations	Recommended	Recommended	Recommended	Optional
Physical Separation	Recommended	Recommended	Recommended	N/A
Assembly Error Mitigation	Recommended for all in-flight non-recoverable assembly errors	Recommended	Optional	N/A
Fail-Safe Interface Designs	Recommended	Recommended	Recommended	Optional
Hazardous Command Protection	Recommended for commands that can damage spacecraft or cause an irreversible event	Recommended for commands that can damage spacecraft or cause an irreversible event	Recommended for commands that can damage spacecraft or cause an irreversible event	Recommended for commands that can damage spacecraft or cause an irreversible event
Spurious Command Protection	Recommended	Recommended	Optional	N/A
Command Lockouts	Recommended - Lockout of command capability not allowed	Recommended - Lockout of command capability not allowed	Recommended - Lockout of command capability not allowed	Recommended - Lockout of command capability not allowed
Conditional Commands	Recommended	Recommended	Recommended	Optional
Single Function Commands	Recommended	Recommended	Recommended	Optional

Table 13. Fault Identification Guidelines

Fault Identification Guidelines 4.1.3.4.5	Class A	Class B	Class C	Class D
Combined FTA/FMEA	Recommended	Recommended	Recommended – to identify the faults	Recommended – to identify the faults

Table 14. Space Vehicle Safety Device Guidelines

Space Vehicle Safety Device Guidelines 4.1.3.4.6	Class A	Class B	Class C	Class D
Battery Over-Charge or Over-Temperature Protection	Recommended	Recommended	Optional	Optional
Battery Depletion Protection (Li-Ion)	Recommended	Recommended	Optional	Optional
Structural Limits Protection	Recommended	Recommended	Optional	Optional
Propellant Depletion Protection	Recommended	Recommended	Optional	Optional
Closable Shields	Recommended - if so equipped	Recommended - if so equipped	Recommended - if so equipped	Optional

Table 15. Command and Control Fault Protection Guidelines

Command & Control Fault Protection Guidelines 4.1.2.4.7	Class A	Class B	Class C	Class D
Critical Commands	Recommended for potential hazardous conditions	Recommended for potential hazardous conditions	Recommended for potential hazardous conditions	Recommended for potential hazardous conditions
Hazardous Constraint Violations	Recommended on-board protection	Recommended on-board protection	Optional	Optional
Critical Constraint Violations	Recommended Ground Protection	Recommended Ground Protection	Optional	Optional
Timers in Critical Applications	Recommended	Recommended	Optional	Optional

Table 16. Design Utilization Guidelines

Design Utilization Guidelines 4.1.3.4.8	Class A	Class B	Class C	Class D
Solar Array Power Margin	Recommended	Recommended	Recommended	Recommended
Battery Capacity Margin	Recommended	Recommended	Recommended	Recommended
Processor Throughput Utilization	Recommended	Recommended	Recommended	Recommended
Processor Memory Utilization	Recommended	Recommended	Recommended	Recommended
Databus Utilization	Recommended	Recommended	Recommended	Recommended
Expendables Margin	Recommended	Recommended	Recommended	Optional

Table 17. Autonomous Fault Protection Guidelines

Autonomous Fault Protection Guidelines 4.1.3.4.9	Class A	Class B	Class C	Class D
Ground Command Response Initiation	Recommended	Recommended	Optional	Optional
Fault Response Enable/Disable Capability	Recommended	Recommended	Optional	Optional
Fault Monitor Enable/Disable Capability	Recommended	Recommended	Optional	Optional
Fault Detection				
Fault Detection - Direct Detection	Recommended	Recommended	Optional	Optional
Fault Detection - Detection Coverage	Recommended	Recommended	Optional	Optional
Response Initiation	Recommended	Recommended	Optional	Optional
Fault Protection Modifications	Recommended	Recommended	Optional	Optional
False Alarm/Spurious Signal Negation	Recommended	Recommended	Optional	Optional
Fault Response Timeliness	Recommended	Recommended	Optional	Optional
Fault Responses False Alarm Tolerance	Recommended	Recommended	Optional	Optional
Return to Previously Swapped Equipment	Optional - recommended default disabled, Ground ability to disable	Optional - recommended default disabled, Ground ability to disable	Optional	Optional

Table 18. Diagnostic Data Guidelines

Diagnostic Data Guidelines 4.1.3.4.10	Class A	Class B	Class C	Class D
Telemetry Coverage	Recommended during all mission Critical Events	Recommended during all mission Critical Events	Recommended during all mission Critical Events	Recommended during all mission Critical Events
Real-Time Telemetry	Recommended	Recommended	Optional	Optional
Stored Telemetry (TLM) for Out Of View (OOV) Operations	Recommended	Recommended	Optional	Optional
Anomaly Telemetry Data Storage	Recommended	Recommended	Optional	Optional
Fault Management Mode Telemetry	Recommended	Recommended	Optional	Optional
Fault Response Diagnostic Data	Recommended	Recommended	Optional	Optional
Preservation of Stored and Diagnostic Data	Recommended - Fault responses must not destroy critical diagnostic data	Recommended - Fault responses must not destroy critical diagnostic data - NASA PD-ED-1243	Optional	Optional
Hardwired Critical Telemetry	Recommended	Recommended	Optional	Optional
Processor Memory Dump Capability	Recommended	Recommended	Recommended	Recommended
Archive Telemetry	Recommended	Recommended	Optional	Optional
Archive Command History	Recommended	Recommended	Optional	Optional

Table 19. Damage Minimization Features Guidelines

Damage Minimization Features 4.1.3.4.11	Class A	Class B	Class C	Class D
Maximize Propellant Load	Recommended	Recommended	Optional	Optional
Re-programmable Processor	Recommended	Recommended	Recommended	Recommended
Polarity Error Correction Capability	Recommended	Recommended	Recommended	Optional
New Technology Safe Guards	Recommended when practical	Recommended when practical	Optional	Optional

Table 20. Fault Management Verification Guidelines

Fault Management Verification 4.1.3.5	Class A	Class B	Class C	Class D
Test Like You Fly	Recommended	Recommended	Recommended	Recommended
Fault management logic paths	Recommended	Recommended	Recommended	Recommended
Verification by Similarity	Recommended	Recommended	Recommended	Recommended
Worst Case Circuit Analysis (WCCA)	Recommended	Recommended	Recommended	Recommended
Threshold, Persistence, Interference (TPI) Analysis	Recommended	Recommended	Recommended	Recommended
Contingency Mode Analyses	Recommended	Recommended	Recommended	Recommended
Safe Mode Analysis	Recommended	Recommended	Recommended	Recommended
Low Power Response Analysis	Recommended	Recommended	Recommended	Recommended
Worst Case Power and Thermal Survival Analysis	Recommended	Recommended	Recommended	Recommended
Steady State Analysis	Recommended	Recommended	Recommended	Recommended
Uplink and Downlink Visibility Analysis	Recommended	Recommended	Recommended	Recommended

8. References

1. Johnson, D. M., (1996) A Review of Fault Management Techniques Used in Safety-Critical Avionic Systems, *Prog. Aerospace Sci.* Vol. 32, pp. 415-431

2. Cheng, P. G. (2007). *Five Common Mistakes Reviewers Should Look Out For.* The Aerospace Corporation, Systems Engineering Division - Risk Assessment and Management Subdivision. El Segundo: The Aerospace Corporation

3. Harland, D. M., & Lorenz, R. D. (2006). Space Systems Failures - Disasters and Rescues of Satellites, Rockets and Space Probes. Chichester, UK: Praxis Publishing

4. 10001_001_1_200902 "Space Vehicle Hazard Analysis", 2009, Thiel, Blanck, Homma"

5. Englehart, W. C. (Ed.). (2005). *Space Vehicle Systems Engineering Handbook.* El Segundo: The Aerospace Corporation. TOR-2006(8506)-4494

6. Aerospace Space Systems Engineering Database, The Aerospace Corporation.

7. *Fault Protection - Preferred Practice No. PD-EC-1243.* National Aeronautics and Space Administration. 1995

8. *Separation and Isolation Guidelines for Printed Wiring Boards Intended for Space Use.* Hogan, S. L. The Aerospace Corporation TOR 2008(8546)-7335

9. Enabling Requirements for Dead Bus Recovery Capability", Landis, Dave: Aerospace Corporation, 2007

10. *RFSS*, August, 2007, R. Ambrose, Lockheed Martin Space Systems

11. Dunn, W. R. (2002). *Practical Design of Safety-Critical Computer Systems.* Slovang, CA, USA: Reliability Press

12. Knight, J. C. and Leveson, N. G. (1986) An experimental evaluation of the assumption of failure independence in multi-version programming. In *IEEE Transactions on Software Engineering,* Vol. SE-12, No.I, January 1986, pp. 96-109

13. *While We Were Sleeping: The Loss of the Lewis* Spacecraft, B. O'Conner, NASA 2007

14. SAE ARP 5580, page 8, Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications

15. JEDEC Publication No. 131A, page 2, Potential Failure Mode and Effects Analysis (FMEA)

16. MIL-STD 1629a, page 4, Procedures for Performing a Failure Mode, Effects and Criticality Analysis

17. NASA Lewis Research Center, Tools of Reliability Analysis—Introduction and FurBas, 2009.

18. Lala, J. H. and Harper, R. E. (1994) Architectural principles for safety-critical real-time applications. In *Proceedings of tile IEEE Vol, 82, No. 1, January 1994* (IEEE), pp. 25-40

19. *NASA Fault Tree Handbook with Aerospace Applications*, NASA Office of Safety and Mission Assurance, August 2002

20. Duphily, R. (2007). *Fault Management ECA Mini Tutorial.* The Aerospace Corporation

21. *Review Findings of the Fault Management Architecture and Implementation for Study C*, D. Byrne, K. Fletcher, S. Hogan, Aug 2007, The Aerospace Corporation, TOR 2007(3907)-7085)

22. 10001_002_1_200902 "Space Vehicle Classification & Mission Assurance Requirements Guide", 2009, Thiel, Blanck"

23. Landis, Dave: "Enabling Requirements for Dead Bus Recovery Capability," Aerospace Corporation, 2007.

24. MIL-STD-1553B, Digital Time Division Command/Response Multiplex Data Bus. United States Department of Defense, September 1987.

25. Marderness, H. P., "Voyager Engineering Improvements for Uranus Encounter", AIAA-1986-2110, in *Astrodynamics Conference*.2New York, American Institute of Aeronautics and Astrodautics, 1986, p. 152-167.

9. Acronyms and Abbreviations

1SOPS	1 Satellite Operations
2SOPS	2 Satellite Operations
ACS	Attitude Control Subsystem
ACU	Attitude Control Unit
AKA	Also Known As
AKM	Apogee Kick Motor
BIT	Built-in-Test
BR	Byzantine Resilience
CAF	Conductive Anodic Filaments
CCD	Charge Coupled Device
CDH	Command Data Handling
CDR	Critical Design Review
CDU	Command and Data Unit
CMF	Common Mode Failure
CMG	Control Moment Gyroscope
CONOPs	Concepts of Operation
CPU	Central Processor Unit
CRC	Cyclic Redundancy Code
CTU	Command and Telemetry Unit
DFH	Dong Fang Hong
DOD	Department of Defense
DOD	Department of Defense
ECA	Effects and Criticality Analysis
EDAC	Error Detection and Correction
EM	Engineering Model
EMC	Electro-Magnetic Compatibility
EMI	Electro-Magnetic Interferance
ESD	Electrostatic Discharge
EUV	Extreme Ultra-Violet
FHA	Functional Hazard Analysis
FM	Fault Management
FMEA	Failure Modes and Effect Analysis
FMECA	Failure Modes, Effects, and Analysis
FMET	Failure Mode and Effects Testing
FPGA	Field Programmable Gate Array
FSW	Flight Software
FT	Fault Tree
FTA	Fault Tree Analysis
GEO	Geosynchronous Earth Orbit
GTO	Geosynchronous Transfer Orbit
HITL	Hardware In-the-loop Testing
HW	Hardware
I&T	Integration and Test
IIR	Block 2 Replacement (GPS)
IMA	Integrated Modular Avionics

IPT	Interference, Persistence, and Thresholds
IR	Infra-Red
IRU	Inertial Reference Unit
IRU	Internal Reference Unit
IUS	Inertial Upper Stage
KIR	Krypto Graphic Interface Receiver
LMMS	Lockheed Martin Missile System
LRM	Line-replaceable modules
MAIW	Mission Assurance Industry Workshop
MCS	Missile Control Station
MTI	Multi-Spectral Thermal Imager
NOAA(I)	National Ocean Graphic and Atmospheric Administration Imaging
NOSO	Navstar Operations Support Office
OOV	Out of view
OPSCON	Operations Concept
PCI	Peripheral Connect Interface
PDR	Preliminary Design Review
PI	Project Investigator
POC	Point of Contact
PRA	Probabilistic Risk Assessment
PROM	Propagated Read-only Memory
REDMAN	Redundancy Management
SBIRS	Space-based Infrared Satellite System
SCP	Spacecraft control Processors
SDR	System Design Review
SECDED	Single-bit Error Correction, Double-bit Error Detection
SEE	Single Event Effect
SEE	Solar EUV Experiment
SEU	Single Event Upset
SMC/CZK	Space and Missle Center / Los Angeles Air Force Base Bldg 120
SOHO	Solar and Heliospheric Obseruatory
SPF	Single-point Failure
SPU	Sensor Processing Unit
SRM	Solid-rocket Motor
SRR	System Requirements Review
ST	Success Tree
SUNSAT	Stellenbosch University Satellite
SV	Space Vehicle
SW	Software
TCAL	Telescopic Calibration
TDRS	Tracking and Data Relay Satellie
TIDI	Timed Doppler Interferometer
TIMED	Thermosphere, Ionosphere, Mesosphere, Energetics, and Dynamics
TIU	Telemetry Interface Unit
TLM	Telemetry
TMR	Triple-modular Redundancy
TOMS-EPS	The Total Ozone Mapping Spectronometer in NASA’s Earth Probe Series
TOR	Technical Operating Report

TPI	Threshold, Persistence, and Interference
TPI	Thresholds, Persistence, and Interference
TTCE	Time to Critical Effect
TTI	Time to Irreversibility
WDT	Watchdog Timers
WIRE	Wide Field Infra-Red Explorer