Failure-Tolerant Avionics for Crewed Space Systems

Recommended Best Practices

Report Approval and Revision History

NOTE: This document was approved at the July 11, 2024, NRB.

Approved:	Timmy Wilson Digitally signed by Timmy Wilson Date: 2024.07.16 12:59:31 -04'00'
	NESC Director

Version	Description of Revision	Office of Primary Responsibility	Effective Date
1.0	Initial Release	Robert F. Hodson, NASA Technical Fellow for Aviation, LaRC	07/11/2024

Signatures

Submitted by:

Robert Hodson Digitally signed by Robert Hodson Date: 2024.07.16 11:46:15 -04'00'

Dr. Robert F. Hodson
NASA Technical Fellow for Avionics

Significant Contributors:

Andrew Loveless Digitally signed by Andrew Loveless Date: 2024.07.16 10:33:56 -05'00'

Dr. Andrew Loveless

Wilfredo Torres-pomales Digitally signed by Wilfredo Torres-pomales Date: 2024.07.16 10:08:17 -04'00'

Mr. Wilfredo Torres-Pomales

PAUL MINER Digitally signed by PAUL MINER Date: 2024.07.16 10:02:32 -04'00'

Dr. Paul S. Miner

Signatories declare the recommendation and rationale compiled in this position paper are factually based from research and literature, program/project experience/documents, and program/contractor meetings and interactions.

Table of Contents

1 Executive Summary

The avionics system in a crewed spacecraft is responsible for performing a variety of safety-critical functions, including controlling the position and attitude of the vehicle in space, detecting onboard failures and performing autonomous recovery operations, activating parachutes and abort systems, and relaying commands from ground operators to the vehicle. The loss or incorrect performance of any of these functions can be catastrophic, resulting in the loss of crew or loss of vehicle. For this reason, it is vital to be disciplined and rigorous when designing and analyzing the avionics system in a crewed vehicle to ensure catastrophic failure conditions are identified and mitigated. Unfortunately, it is not always clear to designers what steps are needed to mature an avionics architecture for human spaceflight and demonstrate it is sufficient. NASA's human-rating requirements offer little guidance for design and implementation of human-rated avionics.

This paper provides an overview of some of the major steps needed to mature and justify the design of an avionics system for crewed spacecraft. It is organized as a collection of artifacts or pieces of evidence that NASA needs to assess the system at design reviews, including a functional failure modes and effects analysis (FFMEA), fault containment region (FCR) definitions, the failure hypothesis, and reliability analysis. This paper is intended as a reference for designers working on NASA crewed spaceflight projects, reliability engineers responsible for avionics system assessments, and program managers wanting to understand what evidence is required at design reviews to ensure crew safety and mission success.

2 Introduction

Human spaceflight is extremely perilous. The failure of a critical spacecraft system in flight can readily result in the loss of the crew or mission. Other serious consequences of system failures include significant financial losses (e.g., the unit cost of one spacecraft can exceed $1B [1]), widespread environmental damage, national embarrassment, and/or the cancellation of future spaceflight missions. As missions increasingly target deep space destinations that cannot be easily supported from Earth, the risk to crew is only increasing.

To reduce these risks, crewed spacecraft are required to undergo *"additional rigor and scrutiny"* as part of the human-rating certification process [2], [3]. This certification provides assurance to NASA managers and program stakeholders that all expectations for crew safety are met. This includes ensuring that design and operational uncertainties are minimized, catastrophic onboard hazards are identified and sufficiently controlled (recognizing that such hazards can be obscured by system complexity [3]), and that all system safety risks are explored and reduced as much as reasonable. Moreover, it is necessary to demonstrate that, to the degree possible, crew members can recover the system after failures occur and the crew can be safely recovered when full system recovery is not possible.

The need for human-rating certification influences many critical subsystems on a spacecraft, especially those that interface directly with the crew (e.g., displays and controls, life support). However, non-crew-facing systems are also impacted. This paper focuses on the avionics system, which includes, among other things, the flight computers, onboard data networks, remote interface units, and sensor and actuator interfaces. The avionics system for a crewed vehicle is required to meet much more stringent failure tolerance and reliability requirements than those for non-crewed vehicles. Moreover, the burden of proof for showing these requirements are met is significantly higher. Evidence of requirements compliance is presented at formal reviews (e.g., the Preliminary Design Review (PDR), the Critical Design Review (CDR)) and must be accepted by NASA managers before fabrication and implementation of the avionics can begin.

Unfortunately, there is a disconcerting trend in which failure tolerance requirements are not sufficiently decomposed, understood, and/or interpreted, resulting in insufficient evidence that an avionics architecture satisfies NASA's mission requirements and safety standards. For example, designers may attempt to reuse the failure tolerance approach from an uncrewed vehicle without performing an analysis demonstrating that adequate hazard controls are in place for crewed missions. Similarly, designers may analyze their system only under benign failure assumptions (e.g., loss of function) rather than considering inadvertent activation or other more dangerous failure modes that have been observed in practice. It is the goal of this paper to describe how to avoid these pitfalls and clarify the expectations for designing and justifying the design of failure-tolerant avionics to ensure crew safety on future missions.

This paper is presented as a checklist describing select key artifacts or evidence needed for NASA to understand the design of a failure-tolerant avionics system and determine its suitability for crewed missions. The artifacts are logically split into three categories. First, *requirements artifacts* ensure the designer's intentions align with NASA's goals. These artifacts include derived functional, performance, and safety requirements for the avionics system, as well as a functional failure modes and effects analysis documenting the impact that general functional failure modes (i.e., loss and malfunction) have on the system's ability to meet those requirements. *Design artifacts* describe the architectural approach and its limitations. These artifacts include an overview of the redundancy scheme; redundancy management strategy; failure detection, masking, and recovery capabilities; the fault containment regions; and any assumed limitations on the failure modes of system components. Finally, *analysis artifacts* provide evidence of the system's ability to meet safety requirements. These artifacts include component failure modes and effects analysis, common cause failure analysis, and reliability analysis.

Note that the goal of this paper is not to advocate for a particular architectural approach. A variety of avionics architectures have been used successfully in crewed spacecraft [4], [5], [6]. Rather, it focuses on how to demonstrate the choice of architecture is appropriate given the mission parameters and requirements. Moreover, this paper is not intended to be exhaustive. Instead, it focuses on those artifacts that are currently viewed as most important to passing design reviews but that are often missing or incomplete in practice.

This paper is intended for avionics, software, and safety personnel working on NASA crewed spaceflight projects. Additionally, it contains important considerations for program managers determining what artifacts or evidence to require at program milestones, and the expected maturity of those artifacts. Note that, while this paper focuses on crewed spaceflight, the core process described may also be applicable to flagship uncrewed missions.

3 Background

3.1 The NASA Program Life Cycle

Before proceeding, it is helpful to briefly outline the steps of the NASA Program Life Cycle [7]. These steps are referenced throughout this paper to illustrate what artifacts should be available at various program milestones and the required maturity of those artifacts.

In summary, every NASA crewed spaceflight program progresses through specific milestones spanning concept formulation, system design, implementation, and verification (a subset of the milestones is included in this document). Each milestone requires designers to demonstrate sufficient progress has been made toward meeting program objectives and requires NASA program approval before the program can continue to the next phase.

- *System Requirements Review (SRR)* – Demonstrates the system's functional, performance, and safety requirements will meet the mission objectives.
- *System Definition Review (SDR)* – Demonstrates the system has a realistic architectural approach that will meet the functional, performance, and safety requirements.

- *Preliminary Design Review (PDR)* – Elaborates on the system's design and demonstrates its appropriateness based on trade studies, analysis, and prototype testing.
- *Critical Design Review (CDR)* – Demonstrates the design is complete and the system is ready to progress to implementation and fabrication.
- *System Acceptance Review (SAR)* – Uses test data and verification artifacts to demonstrate the system implementation is mature before the system is integrated into the spacecraft and shipped to the launch site.
- *Flight Readiness Review (FRR)* – Verifies the flight system and all associated ground hardware, software, and processes are ready for flight.

In general, this paper focuses on artifacts that should be produced in a program's *design phase* (i.e., before PDR and CDR).

3.2 Failure Tolerance Terms

This section briefly defines some of the basic failure tolerance terms used in this paper and indicates where definitions assumed in this paper may conflict with other documents or resources. This paper focuses primarily on tolerating faults due to physical phenomena (e.g., device wear out, radiation) rather than design defects. It therefore uses the standard "hardware-centric" definitions. Terms are listed alphabetically.

- *Availability* – A system or component's[1] ability to provide service. Techniques to increase system availability reduce the probability of system failure due to exhaustion of redundancy (i.e., loss of all copies of a critical component).
- *Common Cause Failure* – The failure of multiple fault containment regions due to a single event, design defect, or other violation of independence assumptions.
- *Coverage* – A general term that refers to the fraction of faults or failures that can be detected, masked, or otherwise "covered" by a particular fault or failure tolerance technique. For example, coverage can be used to describe a checksum's ability to indicate a message was corrupted in transit (e.g., detection coverage), a built-in self-test's (BIST) ability to indicate a device is faulty (i.e., BIST coverage), or a system's ability to tolerate certain failure modes (i.e., failure tolerance coverage).
- *Error* – The discrepancy between the state of a faulty system or component and that of a non-faulty system or component (e.g., flipped bits in memory).

[1] "Component" refers to a single element or constituent part of a system. Typical components of an avionics system include compute nodes, network switches, and power buses.

- *Failure* – The discrepancy between the service provided by a faulty system or component and that described by an agreed upon specification.[2]
- *Failure Detection* – A technique in which failures are observed to identify whether and which components have failed, usually followed by recovery actions.
- *Failure Hypothesis* – Describes the quantity and types of component failures the system is being designed to tolerate, along with any assumptions made about recovery time, fault or failure detection coverage, or fault or failure masking coverage.[3]
- *Failure Masking* – A technique in which a failed component is prevented from exhibiting incorrect behavior to other components.
- *Failure Mode* – Describes the behavior exhibited by a failed component (e.g., the types of errors). These can be categorized many ways (e.g., loss versus malfunction, value versus time domain) and described from either the perspective of the failed component (i.e., what does it do) or other components in the system (i.e., what behavior do they observe).
- *Failure Tolerance* – The ability to withstand component failures to prevent system failure.
- *Fault* – The root cause of a component error or failure (e.g., a damaged circuit trace).
- *Fault Containment Region (FCR)* – A collection of components that share hardware resources and thus can be impacted by a single fault. Note that data interfaces between components are typically not considered shared resources for the purpose of defining FCRs (see Section 4.2.2).
- *Hazard* – A state or set of conditions that could result in adverse consequences to the system or the crew.
- *Integrity* – A system or component's ability to ensure the service it provides is either correct or detectably incorrect. Techniques to increase system integrity reduce the probability of system failure due to a system component performing an incorrect action.
- *Latent Fault* – A fault that has not yet caused a component failure (e.g., a stuck bit in an unused region of memory).
- *Reliability* – The probability that a system does not fail over a given time interval.
- *Time to Effect* – The amount of time between a failure or hazard occurring and it affecting the system or the crew. Failures with a short time to effect require the avionics system to

[2] Note that failures can be permanent, transient, or intermittent (see Section 4.2.3).

[3] The rates of failure modes in the Failure Hypothesis are specified in the Reliability Analysis (see Section 4.3.3), and the output of the Reliability Analysis is needed to justify that the Failure Hypothesis is sufficient.

rely more on failure masking and onboard autonomy versus detection and operator intervention.

Note that this paper avoids using common failure tolerance terms that are easily abused (e.g., "credibility"). For more information on failure tolerance concepts used in avionics, refer to [8], [9], [10], [11], [12], and [13].

4 Artifacts for Assessing a Failure-Tolerant Avionics System

This section describes key artifacts NASA needs to assess the design of an avionics architecture for crewed spaceflight and how to generate them. The avionics in crewed spacecraft are safety-critical, meaning system failure is catastrophic. The purpose of the artifacts is to demonstrate that the risk of such failure of the avionics system is sufficiently low.

The artifacts are split into three general categories: (1) requirements artifacts, (2) design artifacts, and (3) analysis artifacts. For simplicity, these artifacts are described roughly in the order they are produced. However, in many cases it is most natural to develop several artifacts concurrently (e.g., Fault Containment Regions and Failure Hypothesis). Most artifacts should be continuously updated or matured throughout the early stages of the program.

4.1 Requirements Artifacts

4.1.1 Concurrence on and Decomposition of Requirements

At the start of a spaceflight program, NASA provides several types of requirements that are relevant to the design of the avionics system. These include *operational* requirements, *functional* requirements, *performance* requirements, and *safety* requirements. The first step to designing a failure-tolerant avionics system that meets the requirements, and demonstrating it is sufficient, is ensuring that the designers and NASA reach consensus on the meaning of and intent behind the requirements.

Operational, functional, and performance requirements are typically easy to interpret and well-covered by other resources [14]. This paper focuses on the safety requirements. Safety requirements describe the system's ability to continue operating in the presence of off-nominal conditions. NASA typically provides two types of safety requirements:

- *Failure tolerance* requirements specify the minimum number of failed components the system must be capable of operating with. Sometimes different requirements are provided for loss of crew versus loss of mission.[4] When applying these requirements to the avionics system, each avionics element (e.g., compute node, network switch, power bus) is considered a "component." Unless otherwise specified in the NASA requirement, no

[4] Note that [2] ties failure tolerance requirements to catastrophic hazards, which include all hazards that result in loss of crew, and may also include those resulting in loss of mission.

limiting assumptions should be made regarding the boundary of a component,[5] the behavior of failed components, or the simultaneous occurrence of failures. Similarly, the presence of failed components should not be conflated with unavoidable consequences of operating in the space environment. For example, a single failure tolerant requirement does not free designers from needing to tolerate a radiation-induced transient upset after a hardware failure occurs. Note that emergency systems (e.g., launch abort) cannot be used to meet failure tolerance requirements [2]. Further refinements of component failure assumptions can be made, with NASA concurrence, when defining the Fault Containment Regions and Failure Hypothesis (see Sections 4.2.2 and 4.2.3).

- *Reliability* requirements specify the minimum probability that the system does not fail over the mission duration. Again, different requirements may be provided for loss of crew versus loss of mission (where "failure" has different meanings), as well as for different mission phases. Typically, the avionics system is allocated a portion of the system reliability requirement. Simplified reliability models are often used early in a program to allow designers to make architectural trades. Later, higher fidelity models are used to verify the reliability requirement is met subject to the agreed upon Failure Hypothesis (see Section 4.2.3).

Note that the failure tolerance and reliability requirements are designed to complement each other. For example, the failure tolerance requirements ensure designers cannot claim a component is so reliable that it cannot fail. Similarly, the reliability requirements ensure the failure tolerance approach increases the system reliability to an acceptable level. Importantly, this means the reliability requirements may drive the design to tolerate *more* failed components than specified in the failure tolerance requirement.

Once a shared understanding of the failure tolerance and reliability requirements is achieved, designers are responsible for decomposing the requirements into lower-level requirements and allocating them to specific avionic subsystems. These decomposed requirements are refined throughout the early stages of the program. Ample resources are available on the decomposition of requirements (see [15]), so the process is not described here. Once the decomposed requirements are completed, they are presented for NASA concurrence at the System's Requirements Reviews (SRR) and associated meetings.

4.1.2 Functional Failure Modes and Effects Analysis

The next step to developing and justifying the design of a failure-tolerant avionics system is completion of a functional failure modes and effects analysis (FFMEA).[6] A spacecraft's avionics

[5] In other words, designers cannot restrict the meaning of "component" to refer to a single chip or card in an avionics box.

[6] The FFMEA is similar to the functional hazard assessment (FHA) performed in commercial aviation [16].

system must host a variety of functions, including guidance, navigation, and control (GNC); communication; and life support. The FFMEA provides a means of documenting the criticality of each function and using it to inform the choice of failure tolerance approach. Without an FFMEA, there is no basis for discussing whether a failure tolerance strategy is “good enough.”

Note that an FFMEA should be performed in addition to the standard NASA hazard analysis process. Unlike hazard analysis, which starts by identifying top-level hazards and postulating potential hazard causes and controls (i.e., it is top down), the FFMEA (and later FMEA, see Section 4.3.1) work from the bottom up to identify the worst-case consequences of potential failure modes. This practice helps ensure all relevant hazard causes are identified and controlled, which is impractical in complex systems with top-down analysis alone [16].

To complete an FFMEA, designers must compile the list of spacecraft functions the avionics system is required to realize. Then, for each function, designers should document the worst-case effects that general functional failure modes have on the system’s ability to meet its safety requirements (i.e., the potential for loss of crew or mission). At a minimum, two failure modes must be considered in the FFMEA: loss and malfunction. *Loss* is the unexpected absence of the function. *Malfunction* is the arbitrarily incorrect or untimely performance of the function, including inadvertent activation. If the effects of failures depend on the mission phase or the timing of the failure (e.g., time to effect), then this information should be documented in the FFMEA. An example of a simplified FFMEA table entry is shown in Table 1.

Function	**Failure Mode**	**Mission Phase**	**Worst-Case Effects to Mission and Crew**	**Time to Effects**
Control of attitude and pointing	Malfunction	Docking	Collision with visiting vehicle resulting in loss of crew and mission	< 1 minute

Table 1: Example entry from a functional failure modes and effects analysis (FFMEA).

Additionally, the FFMEA should capture the worst-case effects that the *combination* of multiple functional failures has on the system (e.g., loss of functions X and Y). This is because it is possible for a set of functions, which individually are not highly critical, to cause critical effects if they fail simultaneously (or close enough in time that recovery from one functional failure is not possible before another function fails). One example is the simultaneous loss of communication and navigation, which has nearly caused the loss of systems in practice [17]. Note that such combinations should be considered regardless of the number of failures the system is required to tolerate. This is because, if the combination of functional failures has a severe impact, it may drive the need to ensure those failures occur independently in the realized design (see Fault Containment Regions, Section 4.2.2). Moreover, as discussed in Section 4.1.1, the reliability requirements may drive the actual required failure tolerance of the system to be higher than specified in the failure tolerance requirement.[7]

Importantly, there is no notion of failure likelihood in the FFMEA. The goal is to document effects regardless of supposed probability of occurrence. Similarly, there is no consideration of what the architecture may do to mask, respond to, or recover from the described failure modes (i.e., the FFMEA is performed completely independent of the architecture).

It is expected that avionics and software designers may not have enough information to perform the FFMEA in isolation. Thus, completion of the FFMEA will require collaboration with safety engineers and mission planners. A preliminary FFMEA should be presented at or before the System Definition Review (SDR). As the list of spacecraft functions matures, the FFMEA should be updated, with a complete FFMEA presented at the Preliminary Design Review (PDR). Similarly, once the actual avionics architecture is chosen, a Failure Modes and Effects Analysis (Section 4.3.1) should be used to document the impact of failures on the architecture.

4.2 Design Artifacts

4.2.1 Architectural Description

Once the effects of all functional failures are understood, it is possible to design an avionics system that ensures crew safety in the presence of those failures. The purpose of the Architectural Description is to describe the failure tolerance techniques that comprise the avionics architecture and explain how they fit together as part of a comprehensive failure tolerance strategy. The document should make a convincing case to reviewers and program stakeholders that the architecture meets the program's needs. The document should also provide explicit traceability between failures identified in the Functional Failure Modes and Effects Analysis (Section 4.1.2) that could result in loss of crew or mission and how those failures are mitigated by the architecture.

Below are some key pieces of information that should be included in this Architectural Description document.

- *Physical topology*
    - List the components that make up the avionics system, including compute nodes, peripheral cards, network switches, remote interface units, and input/output (I/O) devices.
    - Illustrate the network connectivity between components and document any network protocols used (e.g., MIL-STD-1553, IEEE 802.3).

[7] Recall that the failure tolerance requirement described in Section 4.1.1 is a *floor* and does not mean the system may not end up needing to tolerate more failures to satisfy the reliability requirements.

    - Describe the connectivity from each component to the power subsystem (e.g., which redundant power channel feeds each device).
    - Identify relevant design standards used, if any (e.g., RTCA DO-254).
- *Functional mapping*
    - Describe which components are involved in the performance of each function in the Functional Failures Modes and Effects Analysis (Section 4.1.2) (e.g., Which compute nodes host the GNC software? Which devices are responsible for vehicle health management? Which devices control the activation of pyrotechnics?).
    - Describe the paths that commands take between compute nodes and from compute nodes to actuators.
    - Describe the paths data take from sensors to compute nodes.
- *Time and synchronization*
    - Describe any notions of time onboard, how they are maintained, and how they are correlated with each other (if at all).
    - Describe whether there is any time synchronization between components, what protocols are used for this purpose (e.g., IEEE 1588, SAE AS6802), and what failure tolerance mechanisms are used in these protocols.
    - Describe which hardware components are involved in maintaining synchronization and their roles in the synchronization protocol.
    - Describe the degree to which the correctness of the avionics system depends on any timing assumptions, including the latency of control loops and synchrony of the network.
- *Hardware redundancy and failure independence* – Hardware redundancy is needed to meet NASA's failure tolerance requirements. However, redundancy is only effective if redundant components can be shown to fail independently (i.e., that a single condition or event cannot cause the failure of multiple redundant components).
    - Describe the degrees and types of redundancy used by devices and network segments (e.g., How many redundant components are there? Are they powered or unpowered? Are powered components active or passive?).
    - Describe which aspects of the design ensure failure independence (e.g., Are redundant components electrically isolated or powered by different sources? Are they physically separated? How protected are they from "out of band" failure couplings [18]?).

    - Describe the degree of dissimilarity between redundant components and network segments.

  For more information on failure independence, refer to Fault Containment Regions (Section 4.2.2).

- *Failure masking and detection* – Failure-tolerant systems typically rely on a combination of failure masking and detection. Failure masking uses redundant components, or pieces of information, to prevent failed components from exhibiting incorrect behavior to the rest of the system. Examples include N-majority voting, mid-value selection, self-checking pairs, forward error correction, independent inhibits, and force-sum actuation. In contrast, failure detection strategies attempt to identify and safe failed components before they negatively impact the system but do not attempt to mask them. There are two desirable properties of a failure detection approach: *completeness* (i.e., every failed component is identified) and *correctness* (i.e., no healthy component is identified). In practice, ambiguity of failure signatures typically drives systems to prioritize one property over the other [19]. If not carefully managed, then bias toward completeness can lead to exhaustion of redundancy (i.e., loss of all copies of a critical component). Similarly, bias toward correctness can lead to system failure due to undetected malfunction.

  The choice of masking or detection depends on the time to effect and the consequence of failures (i.e., short time to effect failures often require masking). Both techniques can also be combined. For example, voting systems often use failure detection strategies to identify and isolate failed channels. Masking can also reduce the number of failure modes that system-level failure detection logic must consider (e.g., failure of a self-checking pair manifests as the absence of messages [8]).

    - Describe what failure masking techniques are used in the avionics system (if any) and what failure modes they are intended to mask.
    - Describe what techniques are used to detect and isolate failed components, what failure signatures are assumed to indicate the failure of a component, how the system avoids spurious failure detection/false positives [8] (e.g., how system monitors are used and thresholds are determined), and how the system recovers in cases where spurious failures are detected and acted upon).
    - Describe how the system degrades in response to failures (e.g., does losing a component change the failure tolerance strategy?) and how failed components that have been repaired/replaced are reintegrated into the system.

[8] Avoiding spurious failure detection is important to prevent premature exhaustion of redundancy and minimize distractions for the crew.

    - Describe any strategies for detecting latent faults before failures manifest (e.g., periodic scrubbing of critical inhibits and failure detection mechanisms).

- *Availability mechanisms* – Availability refers to a system or component's ability to provide service (i.e., realize its required functionality) in the presence of failures or faults. Insufficient availability can lead to system failure due to exhaustion of redundancy or inability to detect or mitigate a failure before it results in a catastrophic effect.
    - Describe what mechanisms are used to increase the availability of the system (e.g., hardware redundancy, software redundancy, acknowledgements, retransmissions, erasure coding, and onboard maintenance and replacement).
    - Describe how components down-select commands from redundant components or redundant messages from a single component.
- *Integrity mechanisms* – Integrity refers to a system or component's ability to ensure the service it provides is correct or detectably incorrect in the presence of component failures.[9] Insufficient integrity can lead to system failure due to a component performing an incorrect action (e.g., sending an incorrect command, executing commands out of order) or performing correct actions at the incorrect time or on incorrect data. Techniques that increase integrity include the failure masking techniques outlined earlier and error detection mechanisms like error detection codes (e.g., cyclic redundancy checks), timestamp checks, data sanity checks, digital signatures, and built-in self-tests. They also include temporal redundancy schemes, such as repeated software execution and two-step commanding.
    - Describe what mechanisms are used to increase the integrity of the avionics system (and the components within it) and how these integrity mechanisms interact with the availability mechanisms described previously.
    - Describe how integrity is maintained end-to-end from sensors to compute nodes to actuators.
    - Describe which failure modes the integrity mechanisms can mask or detect and which they cannot, along with the degree of coverage they achieve (see Failure Hypothesis, Section 4.2.3).
- *Onboard autonomy* – Failure-tolerant systems vary in the degree to which they rely on autonomy versus operator interaction to perform onboard functions such as failure detection and recovery. For example, a highly autonomous system may use onboard computers to detect and power down failed components, while other systems may require

[9] "Detectably incorrect" means users of the service can tell the result should not be trusted (e.g., a network packet with an incorrect frame check sequence is detectably incorrect).

ground operators to detect onboard failures using down-linked telemetry. Note that NPR 8705.2C has specific requirements regarding autonomy in human-rated systems, including that (1) spacecraft functions must be able to operate autonomously if their loss could result in a catastrophic event, and (2) human operators (e.g., crew or ground) must be able to monitor and control the system if doing so is needed to prevent catastrophic events [3].

  - Describe the degree to which the architecture relies on autonomy or operator intervention for performance of failure detection, management, or recovery.
  - Describe what protections prevent inadvertent activation of autonomous failure recovery or reconfiguration procedures.
  - Describe what telemetry is available to operators to allow them to monitor the system and which mechanisms ensure the integrity and availability of this telemetry.
  - Describe which mechanisms allow human operators to control the system to prevent catastrophic events.

- *Hierarchy of failure management* – Systems vary in the hierarchy of their failure management strategies. For example, in one system the engine controllers may share engine health and status with a higher-level vehicle manager and rely on the vehicle manager to passivate engine failures. In another system, the engine controllers may mitigate failures locally and simply report to the vehicle manager whether mitigation succeeded. Typically, failures with short time to effect require an increased degree of onboard autonomy and lower-level detection and mitigation.
  - Describe the hierarchy of failure management loops and how it influences the choice of availability and integrity mechanisms described previously.
  - Describe how the locality of failure detection and mitigation functions are influenced by the time to effect of failures in the FFMEA (Section 4.1.2).
- *Initialization, startup, and restart* – During system startup and restart after failure, there is some amount of time during which some components are fully initialized and others are not. During this time, certain failure tolerance approaches and distributed algorithms (e.g., for synchronization) will not function properly (e.g., a voting system may only have one available compute node) [13]. Also, the absence of data from uninitialized components may trigger failure detection logic and cause incorrect system responses (e.g., failover to backup compute nodes).
  - Describe which mechanisms the system uses to ensure correct operation at startup or restart (e.g., definition of default values) and how these mechanisms do not interfere with post-startup operation.

    - Describe whether the failure tolerance strategy is different during startup or restart.

- *Resource utilization and management* – An important aspect of designing a failure-tolerant avionics system is ensuring all onboard functions have sufficient hardware resources (e.g., processing time, buffer space, and network bandwidth). The exhaustion of such resources could cause the system or a component to effectively fail even in the absence of faults, as well as trigger failure detection mechanisms inadvertently. It is also necessary for some mechanisms to enforce limits on resource usage or otherwise overprovision resources to ensure the system operates correctly even if a failed component exceeds its resource allocations.
    - Describe how hardware resources are divided between functions, what mechanisms enforce this division (e.g., ARINC 653 partitioning [20], network partitioning), and identify any key technical performance measures (TPMs) for tracking resource usage.
    - Describe which prioritization and scheduling mechanisms are used to determine access to shared resources, as well as the maximum resource utilizations allowed to ensure system schedulability with these mechanisms.
- *Fault-tolerant hardware* – Some avionic systems use specialized "high-integrity" or internally fault-tolerant components in their designs [21], [22]. The purpose is to restrict the components to more benign failure modes that are easier for the system to tolerate (see Failure Hypothesis, Section 4.2.3) [22], or to allow components to recover from faults internally before they manifest as failures. Such components are common in highly critical subsystems that can directly cause a catastrophic hazard. For example, the autonomous flight termination systems in launchers should contain high-integrity components.
    - Describe any high-integrity or internally fault-tolerant components in the design and the degree to which the correctness of the system relies on them. Any assumed restrictions on the failure modes or increases in the reliability of such devices must be documented in the relevant artifacts (e.g., Sections 4.2.3 or 4.3.3) and proven to be valid.
- *Fundamental algorithms* – Failure-tolerant avionic systems commonly depend on a variety of fundamental algorithms upon which the rest of the failure detection, isolation, and recovery strategy is based. Examples include algorithms for ensuring synchronization between redundant components [23], input agreement on sensor data [24], and that redundant components maintain agreed upon state [25].
    - Describe any fundamental algorithms needed in the architecture and their purpose. If, for any reason, the architecture does not require a typical fundamental algorithm (e.g., replicated state is allowed to diverge), then evidence

of why this practice is acceptable must be provided. For more information on identifying and proving the correctness of key distributed algorithms, refer to Section 4.2.4.

Note that availability and integrity are both important properties of any failure-tolerant avionics system. However, techniques that increase one property often decrease the other. For example, use of a self-checking pair increases the probability that a system's output is correct, thus increasing integrity. However, this also increases the probability that the system produces no output at all (as the failure of either of two components silences the pair), reducing availability. For this reason, it is important that designers consider the tradeoff between availability and integrity when designing their systems. Moreover, the Architectural Description should clearly illustrate how mechanisms to increase availability and integrity are used in combination to control hazards and meet system requirements.

Note that failure tolerance properties are typically not composable. In other words, a system does not become failure tolerant simply because it is made up of failure-tolerant subsystems or components. Therefore, it is important to describe how integration of the various failure-tolerant subsystems/components justifies a claim of system-level failure tolerance. This should be done by putting the system's various failure tolerance strategies in the context of the system's high-level control loops (e.g., sensing, computing, commanding).

Importantly, in addition to simply describing the failure tolerance strategies used in the avionics architecture and how they work together, it is necessary to *formally discharge* the functional failures in the FFMEA (Section 4.1.2) based on the combination of these strategies. At a minimum, this requires both:

- Explaining how the architectural techniques mitigate each of the functional failures in the FFMEA that could result in loss of crew or mission, given the time to effect.
- Providing a data-driven argument for why the identified techniques are sufficient (e.g., based on analysis, additional documentation).

If the provided or derived failure tolerance requirements dictate that the system must tolerate *f* failures, then this process must be repeated for all combinations of *f* failures in the FFMEA that could result in the loss of crew or mission. Moreover, depending on the risk tolerance of the program and the initial requirements, it may be necessary to document how the system mitigates failures that result in less critical effects. Note that the process of discharging failures in the FFMEA must be completed regardless of whether the architecture is based on a heritage design. It is expected that arguments of sufficiency will be based not only on the contents of the Architectural Description but also on the artifacts in the following sections.

The Architectural Description is essential to understanding the system's failure tolerance philosophy and therefore must be available for review at the Preliminary Design Review (PDR). At this stage, it should be possible to demonstrate the sufficiency of the architecture to tolerate critical functional failures in the FFMEA (Section 4.1.2), even if all claims cannot yet be fully substantiated (e.g., because they depend on other, less mature artifacts). By the Critical Design Review (CDR), the Architectural Description should be complete, comprehensive, and fully substantiate the architecture's ability to tolerate failures in the FFMEA according to the program's safety requirements.

4.2.2 Definition of Fault Containment Regions

As part of developing and refining the avionics architecture, it is necessary to document any assumptions made about the independence of failures (e.g., that hardware component X cannot induce the failure of component Y). Such documentation is used as the basis for describing the failure modes the system is designed to tolerate in the Failure Hypothesis (Section 4.2.3). The documentation also informs development of the Reliability Analysis (Section 4.3.3) and is essential to demonstrating the failure tolerance requirements are met.

Assumptions about failure independence are documented using fault containment regions (FCRs). An FCR defines a portion of the system that could be impacted by a single fault. Components in the same FCR typically share resources (e.g., computing hardware, backplanes, power buses, or clocking signals). An example of an FCR is shown in Figure 1.

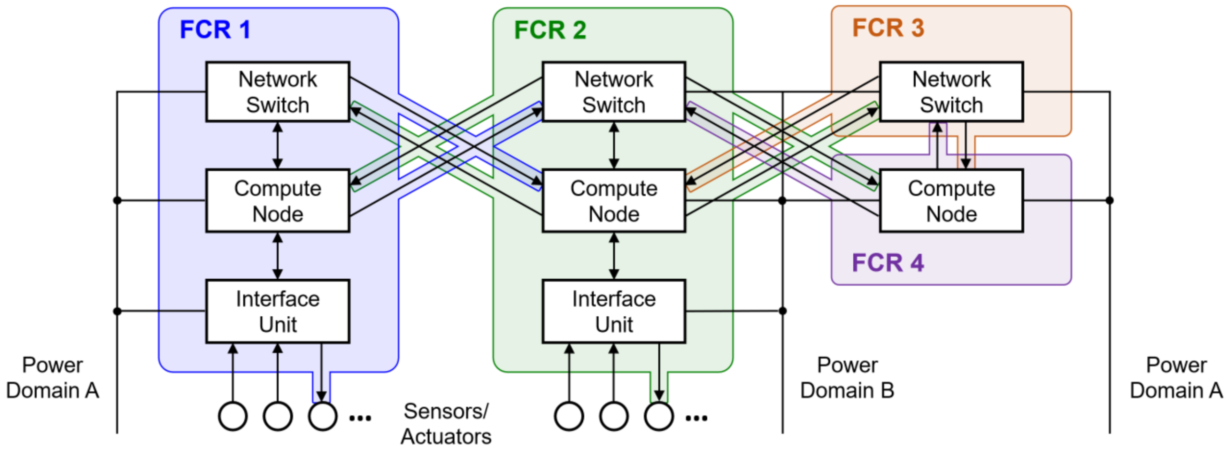

Figure 1: Example of FCRs in a failure-tolerant avionics system. Wires within communication links are typically mapped to FCRs based on the direction/source of the traffic flow.

Because of such shared resources, it is possible for a single fault in an FCR to result in multiple component failures. This can happen due either to (1) a fault in a shared resource directly impacting multiple components or (2) a fault in one component (or an error the fault produces) propagating "backward" *through* a shared resource to a different component. Note that, due to the presence of galvanic isolation and other protection mechanisms in network interfaces, communication links are typically not considered shared resources for the purpose of defining FCRs (i.e., two compute nodes are typically not assumed to reside in the same FCR simply because they communicate over a common link, although sufficient isolation must be demonstrated). A communication link is typically included in the FCR of the closest associated powered device (e.g., a network switch). For bidirectional links, the wires in the links are often assigned to one of two FCRs depending on the direction of traffic flow (see Figure 1).

With this understanding of FCRs, it is possible to further refine the meaning of the failure tolerance requirements described in Section 4.1.1. Specifically, because the failure of different components in an FCR cannot be assumed to happen independently, any number of failures within an FCR should be treated as a *single failure*. In other words, the “component boundaries” referenced in Section 4.1.1 should map to the boundaries of FCRs. A system required to tolerate *f* failed components is in practice required to tolerate *f* failed FCRs, where the number of failed components in a failed FCR is unbounded. This means that all combinations of functions in the FFMEA (Section 4.1.2) whose simultaneous failure could result in loss of crew or mission must be spread across at least *f + 1* FCRs.

In addition to the definition of FCRs, care must be taken to identify any source of *common cause failures* in the system (i.e., circumstances that could result in the failure of multiple FCRs). Sources of common cause failures include but are not limited to 1) common software (i.e., using the same software, frameworks, or compilers in multiple FCRs) [26], 2) common hardware (i.e., using the same hardware components, designs, manufacturing techniques, or calibration processes in multiple FCRs), 3) common physical space (i.e., placing FCRs in the same physical volume such that a single event, such as a fire or solar particle event, could impact multiple FCRs simultaneously), and 4) common external resources (i.e., connecting multiple FCRs to the same external signals or resources, such as clock, reset, or power, without proper isolation/protections). For example, a damaging overvoltage on Power Domain A in Figure 1 could disable FCRs 1, 3, and 4 unless FCRs 3 and 4 are specifically designed to block the overvoltage on Power Domain A and use Power Domain B.

The degree to which the possibility of common cause failures is acceptable depends on the program’s risk posture. In practice, it is not uncommon for some number of sources of common cause failures to exist. Note that external sources of common cause failures may be divided into those that are nominal (e.g., radiation, vibration) and off-nominal (e.g., fire). As part of meeting electrical, electronic, and electromechanical (EEE) parts requirements, designers must often demonstrate components have sufficient resilience to nominal external sources of common cause failures [27].

Note that, although this paper focuses on top level FCRs defined at the system level, FCRs can also be defined hierarchically at the box, board, and component levels. For example, fault-tolerant devices are often designed with internal FCRs intended to prevent faults from propagating between redundant circuit components (e.g., processors, memories, fail-safe inhibits) [28], [29]. The degree to which such hierarchical FCRs are effective depends on the nature and location of the faults. When properly designed, internal FCRs can significantly reduce the probability of a fault-tolerant component exhibiting certain failure modes (see Section 4.2.3).

An overview of the FCRs and all sources of common cause failures should be presented with the Architectural Description (Section 4.2.1) at the Preliminary Design Review (PDR). By the Critical Design Review (CDR), the FCR definitions and common cause failure sources should be mature and agreed upon by NASA managers and all program stakeholders.

4.2.3 Definition of the Failure Hypothesis

Once the fault containment regions (FCRs) are defined, it is necessary to also define the failure hypothesis. System components, and in particular electronics, can fail in many different ways [30], [31]. The purpose of the failure hypothesis is to document which of these failure modes the avionics system is being designed to tolerate. It focuses on failure modes observed at the interfaces between FCRs. An example is shown in Figure 2. Without this information, it is impossible to assess whether the system meets failure tolerance requirements. It is important that the failure hypothesis be carefully examined and agreed upon early in the program, since the occurrence of even a single uncovered failure (i.e., a failure the system is not designed to tolerate) could result in failure of the entire system.

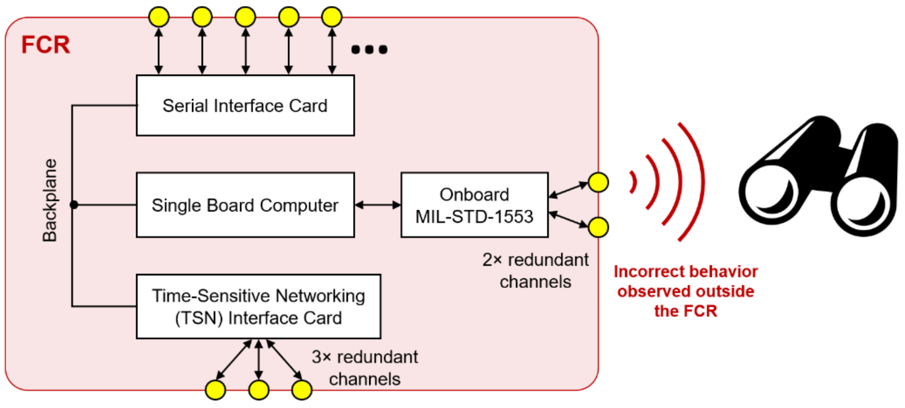

Figure 2: Example FCR containing a single board computer and peripheral cards. The Failure Hypothesis identifies which failure modes the system is designed to tolerate at the interfaces into and out of each FCR (denoted with yellow circles). Failure modes are typically described in terms of the errors that manifest to third-party observers (e.g., corrupted messages).

Failure modes the system can tolerate are expressed using a general failure model. The choice of model is not particularly important, given that the limitations of the architecture are clearly specified and communicated. This paper describes a way to specify failure modes based on work in references [32] and [33]. At the top level, failure modes are split into two categories:

- *Value failures* – Data at the FCR interface are undetectably incorrect (i.e., a *transmissive* failure)[10] or missing (i.e., an *omissive* failure).

[10] 'Undetectably incorrect" is used since detectably incorrect data (e.g., out of range, has a bad checksum) can be ignored and thus treated as an omissive failure.

- *Timing failures* – Data at the FCR interface appear at the wrong time.

Timing failures can be further divided into many subcategories. One special timing failure is *inadvertent activation*, where data are generated (e.g., a command is sent) without the necessary preconditions. Such failures are critical to consider in crewed spacecraft due to their ability to incorrectly activate pyrotechnic or abort systems [29]. Other possible timing failures include *out-of-order failures*, in which data are produced in the wrong order (e.g., perhaps due to a faulty instruction pointer) and *marginal timing failures* [30], in which data are produced slightly too late or early (e.g., in relation to a global synchronized schedule).

All of the previously mentioned failure modes can manifest to other components in one of two ways:

- *Symmetric* – Other system components observe the same error (e.g., a missing message).
- *Asymmetric* – Different system components observe different errors, or some components observe an error and others do not (e.g., only some components receive a message).

Another name for an asymmetric failure is a *Byzantine failure* [30].[11] Each failure mode above can manifest either when data are produced (e.g., a component fails to send some messages) or consumed (e.g., a component corrupts messages it receives). Asymmetric failures also commonly occur in the absence of message drops or corruption (e.g., two components interpret the same message differently due to differences in their local states).

Furthermore, failures can also be classified by their persistence. For example:

- *Permanent* – The failure mode will continue to be observed until the failed component is rebooted, repaired, or replaced by an external entity (e.g., user, another component).
- *Transient* – The failure mode manifests temporarily until the failed component recovers itself (e.g., due to watchdog intervention, memory scrubbing).
- *Intermittent* – The repeated occurrence of a transient failure, usually due to an underlying permanent fault (e.g., repeated message drops at regular intervals).

Note that it is not necessarily the case that an avionics system must tolerate all possible failure modes in all cases. Systems commonly have restrictions on the types and relative timing of failures they tolerate. For example, a system may be designed to tolerate multiple failures, but only if they do not occur simultaneously [34]. Moreover, a system may tolerate multiple benign failures modes (e.g., omissive) but only a single more severe failure (e.g., transmissive, inadvertent activation) [28]. Designers may also make assumptions about which failure modes

[11] Designers sometimes use the term "Byzantine" to refer to any failure that results in arbitrary or erroneous output from a component. This paper instead classifies such failures as (symmetric or asymmetric) transmissive failures.

need to be tolerated depending on where a fault occurs in an FCR. For example, a fault in a computer's network card may be assumed capable of causing a transmissive failure, but a fault in the computer's power supply (or the power bus) may not. The failure hypothesis is the place to document these assumptions.

With that said, the inability of the system to tolerate particular failure modes must be strongly justified. There are two possible types of justification: (1) those based purely on consequence and (2) those based on a combination of consequence and probability. Purely consequence-based justifications must prove the consequence of the failure mode is low enough that all safety requirements can be met without tolerating it (e.g., the failure mode cannot cause loss of crew or loss of mission in any scenario, even in the case the system is required to tolerate multiple failures). This claim must be supported by both the FFMEA (Section 4.1.2) and FMEA (Section 4.3.1). Alternatively, it is necessary to prove that the combination of consequence and probability means the overall risk of not tolerating a particular failure mode is sufficiently low. Possible bases for arguing the low probability of a failure mode include:

- *Hardware architecture* – Some components are designed with internal self-checking or voting mechanisms to mask failures internally and increase the likelihood that failures manifest omissive to other components [22].
- *Safety-layer protocols* – End-to-end error detection codes, sequence number checks, and timestamp checks may increase the likelihood of detecting message corruptions and erroneous messages generated by network components (e.g., switches).
- *Environmental protections* – In the case of failure modes proven to be induced predominately by environmental effects (e.g., radiation, vibration, electromagnetic interference), it may be possible to sufficiently decrease the likelihood of these failure modes by including or bolstering environmental protections in the design (e.g., shielding).
- *Component utilization* – If a component is used only very sparingly, the probability of certain failure modes manifesting during that period may be acceptably low.
- *Detailed Analysis* – A detailed Failure Mode and Effects Analysis (Section 4.3.1) may be able to bound the frequency of a particular failure mode [23].

Note that if a particular failure mode is highly critical (e.g., it can cause a catastrophic hazard with low time to effect), then the system may be required to tolerate it *regardless* of any of the probability-based justifications above. Moreover, with rare exception,[12] failure modes that are excluded from the failure hypothesis should still be represented in the Reliability Analysis (see Section 4.3.3). This practice ensures NASA managers and program stakeholders can still track the

[12] For example, in the case of true self-checking pair hardware [21], the probability of transmissive failures is accepted to be so low (e.g., 10^{-12} failures/hour) that including the possibility of such failures in reliability models has virtually zero impact on the calculated system reliability.

risk of designing the system to not tolerate certain failure modes. This tracking should be performed as part of NASA's standard continuous risk management process [35].

Like the FCRs in Section 4.2.2, a preliminary version of the failure hypothesis should be presented at the Preliminary Design Review (PDR). By the Critical Design Review (CDR), the failure hypothesis should be mature, and any limiting assumptions about component failure behavior should be agreed upon by NASA managers and program stakeholders.

4.2.4 Fundamental Algorithms

Failure-tolerant avionic systems rely on a variety of fundamental algorithms to operate correctly. These algorithms form the "building blocks" upon which the higher-level failure management and failure detection, isolation, and recovery (FDIR) strategies are based. Examples include algorithms for clock synchronization, ensuring input congruency on sensor data, and maintaining agreed upon state on redundant compute nodes [8]. The algorithms specify the steps the system components must perform to maintain certain key system properties.[13]

Definition of fundamental algorithms should begin early in the design cycle before the hardware architecture is finalized. Otherwise, there is risk that the choice of architecture actually *precludes* the definition of an algorithm that meets the failure tolerance requirements given the Failure Hypothesis (Section 4.2.3). In other words, such an algorithm *may not exist* [37], [38], [39]. If such incompatibility is not identified early, then expensive design changes may be required. Key algorithms should also be defined before software development begins. Otherwise, there is a high risk of software errors, since there is no ground truth for the logic the software is supposed to implement. There are many ways to define algorithms, including pseudocode and formal algorithm description languages [40].

Once a fundamental algorithm is defined, evidence should be provided showing the algorithm is correct and relies on valid assumptions. An incorrect algorithm, or an algorithm that makes assumptions that cannot be satisfied in the real system (e.g., synchrony, in-order message delivery), can cause a system to fail even in the absence of any faults or software bugs. Note that software assurance standards and other practices designed to ensure software quality are insufficient for ensuring the correctness of the underlying algorithms [41].

The correctness of algorithms is typically established using proofs. There are a variety of ways to write proofs. For the types of algorithms necessary for avionic systems, simple hand proofs are typically sufficient. Examples of hand proofs can be found in references [42] and [43]. More complex algorithms may require model checkers [44], [45] or even machine-checked proofs [46]. If an algorithm is too complex to prove conventionally, then it may be possible to argue for its correctness based on simulation or significant flight heritage (if the algorithm is completely

[13] Note that an algorithm is distinct from software, just as a blueprint is distinct from a house [36]. An algorithm can be written and proven correct without writing any code to realize the algorithm.

unchanged). However, such techniques cannot ensure correctness under all inputs and thus require NASA agreement to determine if they are sufficient to meet failure tolerance requirements. Ensuring algorithms make valid assumptions typically requires the assumptions be documented and manually reviewed.

Fundamental algorithms in work should be identified at the time of the Preliminary Design Review (PDR). Final algorithm specifications, documentation of underlying assumptions, and proofs (or other evidence) of correctness should be presented at the Critical Design Review (CDR), as they are important preconditions to approving the hardware architecture and proceeding with software development.

4.3 Analysis Artifacts

4.3.1 Failure Modes and Effects Analysis

In the Failure Hypothesis (Section 4.2.3), designers specified the types of component failures the system is being designed to tolerate. However, as the design matures and more information about components becomes available, it is necessary to investigate the actual failure modes of the components, determine whether those failure modes align with the Failure Hypothesis, and assess the degree to which they are mitigated by the realized architecture. This information is typically captured in a Failure Modes and Effects Analysis (FMEA). The FMEA is typically maintained by dedicated reliability engineers[14] and is intended to uncover previously unidentified gaps in the failure tolerance of the system, including single points of failure.

An FMEA typically takes the form of a table. There is a section for each component in the system (e.g., network switch, power distribution unit) and each interface between components (e.g., a backplane). Depending on the maturity of the program or known design shortcomings, it may also be necessary to break sections into subcomponents (e.g., network switch power supply). Each component/interface should have an entry for each possible failure mode of the component. At a minimum, four pieces of information should be recorded per entry. A simplified example of an FMEA entry is shown in Table 2.

- *Failure mode description* – Describe the behavior of the failed component/interface. Failure modes may relate to, for example, loss of function, failure to function at the correct time, degraded performance, performance of the correct function at the wrong time, or performance of the wrong function. Failure modes may manifest in the logical (e.g., data, timing) or physical (e.g., incorrect voltage levels) domains.

[14] Experience shows FMEAs are most effective when performed by independent assessors [16]. Since performing an effective FMEA requires a detailed understanding of the system, it is common to have reliability engineers performing the FMEA embedded in the design team from the start of the program.

- *Failure mode causes* – Describe the possible reasons this failure mode could occur. Consider environmental effects (e.g., radiation, vibration), manufacturing defects, installation or maintenance issues (e.g., strain, cracking), and component deterioration (e.g., random breakdown, wear-out).
- *Failure mode effects* – Describe the potential worst-case effects of the failure mode within the established architecture. These should be broken down into immediate failure effects (i.e., how does the failed component behave), next higher-level effects (i.e., how do components that interact with the failed component react), and system-level effects (i.e., how is the behavior of the system impacted, if at all). Often, these effects are described as though no mitigations were in place. Note that to determine the effects of each failure mode, the functions performed by each component must be known. These were first identified in the FFMEA (Section 4.1.2).
- *Failure mode mitigations* – Describe how the system mitigates or responds to the failure, including failure prevention (e.g., component selection, maintenance, and scrubbing), failure masking or detection strategies, software responses, and system-level corrective actions (e.g., reconfiguration or re-prioritization of mission objectives). Sometimes, these are broken out into different columns in the FMEA. Additionally, describe the amount of time required for mitigations to take effect.

Failure Mode Description	**Failure Mode Causes**	**Failure Mode Effects**	**Failure Mode Mitigations**
Altimeter 1 provides erroneous data to the remote interface unit it is connected to (RIU 1) over RS-232.	Manufacturing defects in, wear-out of, or improper installation of Altimeter 1's optics, internal components (e.g., laser diodes), or serial interface, radiation-induced single-event effects, radiation-induced total dose effects, overvoltage or undervoltage.	RIU 1 receives incorrect data from Altimeter 1, packages it into an Ethernet packet with data from other types of sensors, and sends the packet to the three flight computers (FC1-FC3) over redundant Ethernet networks. If FC1-FC3 acted upon the incorrect RIU1 data, then it could cause loss of vehicle command and control, resulting in loss of mission, loss of vehicle, and/or crew injury or death.	There are three redundant altimeters connected to three redundant RIUs that send data to the three FCs. FC1-FC3 execute a two-round exchange to ensure bitwise agreement on the redundant RIU data, then execute a mid-value selection to choose which altimeter's data are used as input to the GNC software. This process ensures the chosen altimeter data either 1) originates from a non-faulty altimeter, or 2) are bound by data from non-faulty altimeters. Thus, erroneous altimeter data are not acted upon.

Table 2: Example FMEA entry for a laser altimeter that provides data to flight computers in a lander spacecraft's avionics system.

Additionally, FMEAs can be extended to capture criticality. For example, *Failure Mode Effects and Criticality Analysis (FMECA)* includes the consequence, likelihood of occurrence, and likelihood of detection/masking for each entry. These metrics are combined to form a risk score that can be used to prioritize where additional mitigations are necessary.

Identification of failure modes should come from engineering judgment and operational experience with similar devices. Failure mode data should also be "rolled up" from component manufacturers, who often have knowledge of a component's development history and design internals that may point to hard-to-identify failure modes. Finally, identification of failure modes may come from detailed analysis and testing of the component. Note that the amount of component analysis/testing required depends on the contents of the Failure Hypothesis. For example, if the Failure Hypothesis states the system cannot tolerate a certain failure mode that could cause a critical effect (e.g., loss of crew or loss of mission), then *significant effort* should be spent trying to identify scenarios in which such failure modes could occur.

Importantly, regardless of where the program is in the development cycle, it is always possible that findings in the FMEA could drive changes to the design. This possibility is a strong motivator for designing the system to tolerate as many failure modes as possible. It is also a motivator for starting the FMEA process early and with continual input from the design team.

Note that the FMEA is a living document that matures throughout the program life cycle. An initial version of the FMEA that is consistent with the Failure Hypothesis should be available at the Critical Design Review (CDR). As the program continues through implementation and test, the FMEA should become significantly more specific and detailed. Ideally, the FMEA should be one of the last documents delivered before launch and include references to all verifications performed to show that critical effects identified in the FMEA were mitigated.

4.3.2 Common Cause Failure Analysis

The failure tolerance of an avionics system relies critically on assumptions about failure independence. For example, that a single event or circumstance cannot result in the failure of multiple redundant components (or specifically, fault containment regions (FCRs). In Section 4.2.2, designers documented sources of common cause failures (CCFs) in the avionics system that could violate these independence assumptions. Typically, NASA must review and approve these known sources of CCFs at the Critical Design Review (CDR). However, as the avionics architecture matures and is integrated with other vehicle subsystems, new sources of common cause failure may emerge. The *Common Cause Failure Analysis (CCFA)* is used to examine the avionics in its vehicle-level and operational contexts to identify and assess the risk of previously unknown sources of CCFs. Like the FMEA, the CCFA is typically performed by independent safety and reliability engineers and evolves throughout the program lifecycle.

The CCFA should be as comprehensive as possible, considering errors (in requirements, design, implementation, manufacturing, installation, operation, and maintenance), hardware failures, environmental factors, and sources of failures external to the avionics system and spacecraft. Commonly, CCFA is performed from three different perspectives: (1) *zonal safety analysis*, to examine the avionics for hazards related to physical colocation and installation in the spacecraft, (2) *particular risk analysis*, to analyze specific threats (e.g., fire, high-energy devices, high-pressure containers, high-intensity electromagnetic radiation emitters, and fluid leaks), and (3) *common mode analysis*, to examine postulated failure causes (e.g., design defects, software commonality), couplings, and their effects on safety. All potential coupling factors and mechanisms of effect propagation should be considered, including data/information flows and mechanical, thermal, and electrical processes.

CCFs should be assessed based on likelihood and consequence. The likelihood of CCFs depends on a variety of factors, including physical separation and segregation; dissimilarity; complexity; environmental controls; maintenance and operational procedures; competence of development, maintenance, and operation personnel; and testing and operational history. The likelihood of CCFs should be assessed using fault tree analysis or other specialized techniques [47]. The consequence should be assessed using generic tools (e.g., FMEA (Section 4.3.1) and fault tree analysis) and specialized tools for causal threat and effects propagation analysis. Both the likelihood and consequence should be represented in the Probabilistic Risk Assessment (Section 4.3.4) to determine the overall risk that CCFs pose to mission success and safety. For example, CCFA is commonly used to calculate the alpha and beta factors used in the PRA [48]. For more information on CCFA methods and tools, see references [49], [50], [51], [52], and [47].

Like the FMEA, an initial version of the CCFA should be available at the Critical Design Review (CDR). At this time, the CCFA should be consistent with all CCFs that were documented with the Fault Containment Regions in the avionics system (Section 4.2.2). The CCFA should be continually updated and assessed against the architecture as the architecture matures and the program continues through implementation and verification.

4.3.3 Avionics Reliability Analysis

This section describes how to demonstrate NASA's reliability requirements are met with respect to the avionics system. Unlike failure tolerance, it is not possible to assess a system's reliability through manual analysis of the architectural description, failure hypothesis, fault containment regions (FCRs), and algorithm specifications. Instead, reliability calculations must be performed using mathematical tools operating on reliability models.

A variety of reliability modeling approaches exist and can potentially be used, including fault tree analysis, reliability block diagrams, and Markov modeling [8], [53]. Regardless of the approach used, the model should include all failure modes documented in the Failure Hypothesis (Section 4.2.3), along with corresponding failure rates. Additionally, the model should include "death states" [8] representing all conditions under which the system may fail and cause loss of crew or mission, including loss of availability (e.g., exhaustion of redundancy) and lack of integrity (e.g., violation of the Failure Hypothesis – covered in more detail in this section).[15]

Failure rates should ideally come from the actual failure history of the same or similar components in a similar spaceflight environment. Alternatively, failure rates should be well-justified by test data. If failure rate data for certain failure modes are not available, then those failure modes should be assumed to manifest with a failure rate that is a nonzero fraction of the aggregate component failure rate (e.g., X% of failures exhibit the failure mode). Due to the difficulty of getting good data on all failure modes, it is good practice to overapproximate the proportion of failures that manifest with severe failure modes in the model [54]. For example, assume 10% of failures occur in the value domain, or are asymmetric. This practice helps ensure the reliability of the architecture is not overly dependent on possibly incorrect assumptions about the unlikelihood of a component exhibiting certain failure modes.[16]

Unless otherwise specified in NASA's initial requirements, the failure rates should account for the possibility of radiation-induced single event effects on system components (e.g., SEUs, SETs) in the spaceflight environment. If not possible, then such single event effects must be represented in the Probabilistic Risk Assessment (Section 4.3.4). In contrast, the accumulative degradation of system components due to long-term radiation exposure (e.g., TID, DDD) is typically not represented in the reliability models but rather is assessed with dedicated parts-wise radiation analysis.[17]

Assumptions about failure independence should be consistent with the Fault Containment Regions (FCRs) defined in Section 4.2.2. For example, failures in different FCRs should be modeled to occur independently, and one fault should be capable of inducing any number of failures in an FCR (subject to any limitations declared in Section 4.2.3).

Importantly, the model should also represent the probability that the Failure Hypothesis (Section 4.2.3) *is violated*. For example, if the architecture relies on cyclic redundancy checks (CRCs) to detect message corruptions caused by network components (e.g., switches), then the possibility that such corruptions escape detection (i.e., a corrupt message has a valid CRC) should be included in the model. The coverage that can be assumed depends on the integrity mechanism used (e.g., 10^{-7} to 10^{-8} undetected corrupt messages/hour may be a reasonable estimate for CRCs

[15] A common deficiency in reliability models is only representing the possibility of system failure due to loss of availability. However, the possibility of failure modes the system is not designed to tolerate can have a larger impact on system reliability.

[16] Another method is to start with the overall device failure rate. Then, subtract the rates of failure modes that can be accurately estimated. Lastly, assign the resulting rate to *each* of the remaining failure modes. The sum of the rates will be greater than the overall device failure rate, but this is better than underestimating.

[17] Components are selected that can withstand total doses that exceed the required mission time.

[55]).[18] Including such probabilities in the model is essential, since even if the probability that a certain failure mode escapes detection is low in *isolation*, repeated application of the same mechanism system-wide can significantly reduce system reliability.[19]

The model should also represent the imperfect coverage of any failure detection mechanisms (i.e., the nonzero probability that a faulty or failed component cannot be detected). This probability depends on the detection mechanism used. For example, the detection coverage of built-in self-tests (BISTs) and watchdog timers is typically assumed to be under 50% [56]. The model should also represent the possibility of false positives in the failure detection logic (e.g., a failed computer incorrectly indicts a healthy component), since such false positives can often trigger a system reconfiguration process (e.g., failing over to a backup flight computer) that makes the system more susceptible to failure. Similarly, such false positives can lead to system failure due to exhaustion of redundancy (i.e., unavailability). Figure 3 shows the significant impact that coverage assumptions have on reliability results.

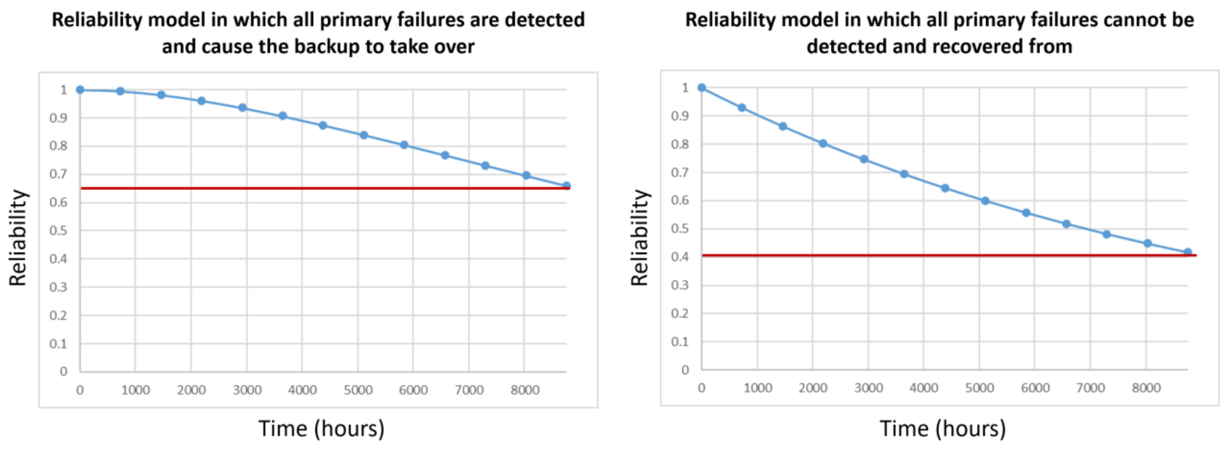

Figure 3: Comparison of the results of two reliability models. Both model a simple failure-tolerant system with two computers, a primary and a backup. One model assumes all failures of the primary cause the backup to take over (left). The other represents the fact that the backup cannot detect all primary failures (right). The left model predicts a 62% higher reliability over one year.

Reliability calculations should be performed over the entire mission length unless otherwise agreed upon.[20] If repair or replacement of failed devices is needed to meet the reliability

[18] Claims of better CRC coverage typically rely on the assumption of random independent bit errors, which does not hold in networks with active inter-stages (e.g., network switches) [55].

[19] An exception can be made for failure modes that can be proven to be *exceedingly* rare (e.g., with rates $<10^{-11}$ failures/hour [22]), since excluding them typically has an insignificant impact on calculated system reliability and reduces the complexity of the model.

[20] For avionics that can be repaired or replaced in orbit, NASA may agree to the use of time intervals shorter than the mission length for the reliability analysis (e.g., 1 year).

requirements, then that information should be clearly communicated with NASA, and repair intervals should be explicitly included in the model.

Ideally, reliability analysis should begin early in a program's life cycle to support architectural trades (i.e., before the System Definition Review (SDR)). At a minimum, a preliminary version of the reliability analysis should be presented at the Preliminary Design Review (PDR). The final version, including all failure modes and detection/masking coverage numbers as described earlier, should be presented at the Critical Design Review (CDR).

4.3.4 Spacecraft Probabilistic Risk Assessment

In addition to performing an Avionics Reliability Analysis (Section 4.3.3), designers and reliability engineers are responsible for supplying data to enable NASA to complete a spacecraft Probabilistic Risk Assessment (PRA). This data includes the Architectural Description (Section 4.2.1), as well as the parameters used in the reliability analysis (e.g., failure rates, failure modes, and failure masking/detection coverage numbers). NASA human-rating certification requirements dictate that NASA establish loss of crew and loss of mission requirements that are verified by a NASA PRA for all crewed spaceflight programs [57].

There are several key differences between the reliability analysis described previously and the PRA performed by NASA. These include:

- The PRA identifies the likelihood of specific negative outcomes (e.g., loss of crew, loss of vehicle, or loss of mission).
- The PRA is holistic, considering all onboard subsystems (not just avionics), as well as possible crew and ground responses to hazards (e.g., crew evacuation via a lifeboat).
- The PRA uses uncertainty distributions for all model parameters/probabilities (e.g., failure rates). More uncertainty is assigned to lower probabilities and probabilities that are not justified by actual operational history.
- The PRA includes the possibility of system failure due to common cause failures (CCFs). These are typically not considered in early reliability analyses.[21] Sources of CCFs in the avionics system were documented in Section 4.2.2. Additional sources of CCFs are identified and analyzed as part of the CCFA (Section 4.3.2).

The PRA evolves throughout the program's life cycle, even continuing to be refined after launch to reflect updates or improvements in both onboard hardware and crew and ground processes. However, data on the avionics to supply the PRA should start being provided early in the program, including initial failure rate, mode, and coverage data. This PRA input data should be consistent

[21] While it is possible to include CCFs in early reliability analyses, the corresponding parameters are often rough approximations. There is a risk that such terms can mask meaningful differences between avionic architectures when performing architectural trades.

with the Reliability Analysis (Section 4.3.3) and integrated into the PRA by the Preliminary Design Review (PDR). Updated data should be provided with the updated Reliability Analysis by the Critical Design Review (CDR). The PRA should also continue to be informed by the CCFA (Section 4.3.2) as it evolves.

5 Conclusion

Avionic systems in crewed spacecraft are responsible for a variety of critical functions whose loss or malfunction can be catastrophic. As a result, an extremely high level of thoroughness and scrutiny is required when designing and analyzing these systems. Despite this, it is not uncommon for the required analysis or design artifacts to be absent or completed inadequately at the time of major program milestones (e.g., the PDR and CDR). This paper provides an overview of some of the major artifacts or pieces of information that NASA requires to assess the failure tolerance of an avionics system for crewed spacecraft. Additionally, it describes the required maturity of these artifacts at design reviews. For convenience, this information is summarized in Table 3. This paper should be used as a reference by both designers working on crewed spaceflight programs and program managers required to assess the maturity and safety of avionic architectures for crewed space systems. It may also be helpful to reliability engineers developing products and performing assessments related to avionic systems.

Artifact/Analysis	**SRR**	**SDR**	**PDR**	**CDR**	**SAR**	**FRR**
Requirements Decomposition (Section 4.1.1)	C					
Functional Failure Modes and Effects Analysis (Section 4.1.2)		P	C			
Architectural Description (Section 4.2.1)			P	C		
Fault Containment Regions (Section 4.2.2)			P	C		
Failure Hypothesis (Section 4.2.3)			P	C		
Fundamental Algorithms (Section 4.2.4)			P	C		
Failure Modes and Effects Analysis (Section 4.3.1)				P	→	C
Common Cause Failure Analysis (Section 4.3.2)				P	→	C
Avionics Reliability Analysis (Section 4.3.3)			P	C		
Probabilistic Risk Assessment Inputs (Section 4.3.4)			P	→	→	C

Table 3: Icons indicate at what stages of the NASA program life cycle the preliminary version of each artifact should be made available (P), and the final version should be completed and agreed upon with NASA managers and program stakeholders (C). Arrows indicate that the artifact is continually updated as new information becomes available. Stages of the program life cycle are described in Section 3.1.

6 Acknowledgments

The authors would like to thank the following individuals, whose detailed reviews and insightful comments greatly improved the quality of this paper:

- Brendan Hall, NASA Johnson Space Center, Jacobs
- Clint Baggerman, NASA Johnson Space Center
- David K. Rutishauser, Ph.D., NASA Johnson Space Center
- Derek Otermat, Ph.D., NASA Kennedy Space Center
- J. Michelle Edwards, Ph.D., NASA Johnson Space Center, Jacobs
- Jon B. Holladay, NASA Goddard Space Flight Center
- Joseph Busa, TTTech North America
- Kevin Driscoll, NASA Johnson Space Center, Jacobs
- Lorraine E. Prokop, Ph.D., NASA Johnson Space Center
- Marc Butler, NASA Kennedy Space Center
- Mark Terrone, NASA Kennedy Space Center
- Steven J. Gentz, NASA Marshall Space Flight Center
- Teri Hamlin, NASA Johnson Space Center
- William T. Smithgall, TTTech North America
- Yuan Chen, Ph.D., NASA Langley Research Center

7 References

[1] E. Howell and D. Dobrijevic, "James Webb Space Telescope (JWST) — A Complete Guide," *SPACE.com*, Apr. 2023. [Online]. Available: https://www.space.com/21925-james-webb-space-telescope-jwst.html

[2] "NASA Technical Requirements for Human-Rating," National Aeronautics and Space Administration, NASA-STD 8719.29, Dec. 2023.

[3] "Human-Rating Requirements for Space Systems," National Aeronautics and Space Administration, NPR 8705.2C, Jul. 2017.

[4] M. Fletcher, "Progression of an Open Architecture: From Orion to Altair and LSS," Honeywell, International, S65-5000-20–0, May 2009.

[5] J. Hanaway and R. Moorehead, "Space Shuttle Avionics System," National Aeronautics and Space Administration, NASA SP-504, Jan. 1989.

[6] J. Graf, C. Reimers, and A. Errington, "ESA's Data Management System for the Russian Segment of the International Space Station," *Eur. Space Agency ESA Bull.*, Feb. 1998.

[7] "NASA Space Flight Program and Project Management Requirements," National Aeronautics and Space Administration, NPR 7120.5F, Aug. 2021.

[8] R. Butler, "A Primer on Architectural Level Fault Tolerance," National Aeronautics and Space Administration, NASA/TM-2008-215108, Feb. 2008.

[9] R. Butler and S. Johnson, "The Art of Fault-Tolerant System Reliability Modeling," National Aeronautics and Space Administration, NASA-TM-102623, Mar. 1990.

[10] W. Torres-Pomales, "Selecting an Architecture for a Safety-Critical Distributed Computer System with Power, Weight and Cost Considerations," National Aeronautics and Space Administration, NASA/TM-2014-218242, Apr. 2014.

[11] A. Avizienis, J.- Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[12] J.-C. Laprie, "Dependability -- Its Attributes, Impairments and Means," in *Predictably Dependable Computing Systems*, in ESPRIT Basic Research Series. Springer Berlin Heidelberg, 1995.

[13] B. Hall and K. Driscoll, "Distributed System Design Checklist," National Aeronautics and Space Administration, NASA/CR–2014-218504, Jul. 2014.

[14] "Technical Requirements Definition." [Online]. Available: https://www.nasa.gov/reference/4-2-technical-requirements-definition/

[15] “Logical Decomposition (of Technical Requirements).” [Online]. Available: https://www.nasa.gov/reference/4-3-logical-decomposition/

[16] C. Spitzer, *The Avionics Handbook*. CRC Press, 2001.

[17] “Stealth Fighters Hit by Software Crash,” *itnews*, Feb. 2007. [Online]. Available: https://www.itnews.com.au/news/stealth-fighters-hit-by-software-crash-74081

[18] K. Driscoll, “Murphy Was an Optimist,” in *Proc. SAFECOMP*, Vienna, Austria, Sep. 2010.

[19] K. Shin and P. Ramanathan, “Diagnosis of Processors with Byzantine Faults in a Distributed Computing System,” in *Proc. 17th Fault Tolerant Computing Symposium (FTCS)*, Pittsburgh, PA, USA, Jul. 1987.

[20] “Avionics Application Software Standard Interface,” Aeronautical Radio, Incorporated (ARINC), ARINC 653, Nov. 2021.

[21] K. Hoyme and K. Driscoll, “SAFEbus,” in *Proc. DASC*, Seattle, WA, USA, Oct. 1992.

[22] K. Driscoll and B. Hall, “Application Agreement and Integration Services,” National Aeronautics and Space Administration, NASA/CR–2013-217963, Feb. 2013.

[23] J. H. Lala and R. E. Harper, “Architectural Principles for Safety-Critical Real-Time Applications,” *Proc IEEE*, no. 1, Jan. 1994.

[24] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Trans Program Lang Syst*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[25] Wilfredo Torres-Pomales, Mahyar R. Malekpour, and Paul S. Miner, “ROBUS-2: A Fault-Tolerant Broadcast Communication System,” NASA Langley Research Center, NASA/TM-2005-213540, Mar. 2005.

[26] L. Prokop, “Considerations for Software Fault Prevention and Tolerance,” National Aeronautics and Space Administration, NESC Technical Bulletin No. 23-06, Sep. 2023.

[27] “Electrical, Electronic, and Electromechanical (EEE) Parts Assurance Standard,” National Aeronautics and Space Administration, NASA-STD-8739.10, Jun. 2017.

[28] A. Loveless, “Notional 1FT Voting Architecture with Time-Triggered Ethernet,” Houston, TX, USA, Nov. 07, 2016. Accessed: Jan. 02, 2019. [Online]. Available: https://ntrs.nasa.gov/citations/20170001652

[29] T. Evans, C. Iannello, and R. Hodson, “Including Key Design Features in Safety-Critical Pyrotechnic Firing Circuits,” National Aeronautics and Space Administration, NESC Technical Bulletin No. 23-01, Mar. 2023.

[30] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg, “Byzantine Fault Tolerance, From Theory to Reality,” in *Proceedings of the 22nd International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Edinburgh, Scotland, Sep. 2003.

[31] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed. Springer US, 2011.

[32] M. H. Azadmanesh and R. M. Kieckhafer, "Exploiting Omissive Faults in Synchronous Approximate Agreement," *IEEE Trans. Comput.*, vol. 49, no. 10, pp. 1031–1042, Oct. 2000.

[33] H. Kopetz, "On the Fault Hypothesis for a Safety-Critical Real-Time System," in *Proc. Automotive Software–Connected Services in Mobile Networks: First Automotive Software Workshop*, San Diego, CA, USA, Jan. 2004.

[34] C. Kouba, Deborah Buscher, and Joseph Busa, "The X-38 Spacecraft Fault-Tolerant Avionics System," in *Proc. MAPLD*, Washington, DC, USA, Aug. 2003.

[35] H. Dezfuli, A. Benjamin, C. Everett, G. Maggio, and M. Stamatelatos, "NASA Risk Management Handbook," National Aeronautics and Space Administration, NASA/SP-2011-3422, Nov. 2011.

[36] L. Lamport, "Thinking Above the Code," Microsoft Research, Jul. 2014.

[37] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *J ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[38] F. Ellen, "Impossibility Results for Distributed Computing Part I," University of Toronto.

[39] N. Lynch, "A Hundred Impossibility Proofs for Distributed Computing," Massachusetts Institute of Technology, MIT/LCS/TM-394, Aug. 1989.

[40] L. Lamport, "The PlusCal Algorithm Language," *Theor. Asp. Comput. ICTAC*, Aug. 2009.

[41] R. W. Butler and G. B. Finelli, "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software," *IEEE Trans. Softw. Eng.*, vol. 19, no. 1, Jan. 1993.

[42] A. Loveless, R. Dreslinski, B. Kasikci, and L. T. X. Phan, "IGOR: Accelerating Byzantine Fault Tolerance for Real-Time Systems with Eager Execution," in *Proc. RTAS*, Nashville, TN, USA, May 2021.

[43] L. Gong, P. Lincoln, and J. Rushby, "Byzantine Agreement with Authentication: Observations and Applications in Tolerating Hybrid and Link Faults," *Dependable Comput. Crit. Appl.*, vol. 10, pp. 139–157, Sep. 1995.

[44] S. Bensalem et al., "An Overview of SAL," in *Proc. Fifth NASA Langley Formal Methods Workshop*, Williamsburg, VA, Jun. 2000.

[45] G. Holzmann, "The Model Checker SPIN," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, May 1997.

[46] D. Cousineau et al., "TLA+ Proofs," in *Proc. International Symposium on Formal Methods*, Paris, France, Aug. 2012.

[47] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, “Fault Tree Handbook with Aerospace Applications,” National Aeronautics and Space Administration, Aug. 2002.

[48] V. Hassija, S. Kumar Chandran, and K. Velusamy, “A Pragmatic Approach to Estimate Alpha Factors for Common Cause Failure Analysis,” *Annals of Nuclear Energy*, vol. 63, 2014.

[49] “Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment,” SAE International, ARP4761A, Dec. 2023.

[50] “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 6: Guidelines on the Application of IEC 61508-2 and IEC 61508-3,” International Electrotechnical Commission, IEC 61508-6, Apr. 2010.

[51] “Reliability and Maintainability (R&M) Standard for Spaceflight and Support Systems,” National Aeronautics and Space Administration, NASA-STD-8729.1A, Sep. 2017.

[52] “Space Vehicle Failure Modes Effects and Criticality Analysis (FMECA) Guide,” The Aerospace Corporation, Aerospace Report No. TOR-2009(8591)-13, Jun. 2009.

[53] R. Butler, “The SURE Approach to Reliability Analysis,” *IEEE Trans. Reliab.*, vol. 41, no. 2, Jun. 1992.

[54] P. Miner, M. Malekpour, and W. Torres-Pomales, “A Conceptual Design for a Reliable Optical Bus (ROBUS),” in *Proc. Digital Avionics Systems Conference (DASC)*, Irvine, CA, USA, Oct. 2002.

[55] M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico, and P. Koopman, “Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems,” in *Proc. DSN*, Yokohama, Japan, Jun. 2005.

[56] A. El-Attar and G. Fahmy, “A Study of Fault Coverage of Standard and Windowed Watchdog Timers,” in *Proc. IEEE International Conference on Signal Processing and Communications*, Dubai, United Arab Emirates, Nov. 2007.

[57] “Crewed Deep Space Systems Human Rating Certification Requirements and Standards for NASA Missions,” National Aeronautics and Space Administration, HEOMD-003, Nov. 2021.