Software Common Mode Failure Incidence

September 2022 – Software F2F

Lorraine Prokop

Software Common Cause Failure - Software Fault Tolerant Design

- What is Software "Common Cause" or "Common Mode" Failure?
  - In many avionic architectures, hardware replication into multiple "strings" is done for hardware fault tolerance
    - Common Example: Three or four identical processors running "lock step" or synchronized vote on their outputs
  - However, in many architectures, the *same software load* runs on these multiple processors
  - Since the same software executes on redundant computers, a software failure normally would affect all strings in the same way at the same time
  - If only one processor is used, then *any* software failure could be considered "common mode" or "common cause"
- Motivation
  - Safety and mission-critical systems should employ software fault tolerance
  - With increasing automation, how can you determine if the software has had an error?
  - Very little literature exists characterizing software errors in real-time avionic systems as a guide on how to best implement fault tolerance

Software Common Cause Classes

- Two classes of software common cause:
  - Fail Silent – Computers all stop outputting, Ex: simultaneous crash
  - Erroneous output – Software does the wrong thing
    - Broader class of errors that can manifest in loss of control
  - Both should be considered when designing for fault tolerance
- Why differentiate?
  - It is much easier to determine of the software has stopped or crashed
    - Most systems implement a watchdog timer
  - How can you determine if the automation or software is doing something *wrong*?
    - Implement an independent monitoring algorithm – what if *that* is wrong (?)
  - Current space systems approach these manifestations in different ways
    - Most use humans in the loop to monitor and determine if the software is behaving as expected

Software Failure Categories

- Fail-Silent Cause Examples (*loss of output*)
  - Operating System Simultaneous Halt
    - Memory access violations, arithmetic errors, memory leaks, coding/logic errors
  - CPU Hog – Process Starvation
    - Infinite loop, priority inversion, extreme latencies
- Erroneous Output Causes Examples (*wrong output*)
  - Coding/Logic Error
    - Missing or Wrong Requirements (many errors can be discussed as “missing” requirements)
    - Insufficient modeling of real-world
    - Latencies/Timing Errors, Task Overruns, real-time behavior misunderstood, not handled
  - Data Parameter Misconfigured
    - Wrong data input, database, Units, precision, sign
  - Version Management Error
    - Software configuration management, software or data version
  - Unanticipated / Erroneous Input
  - Operator / Procedural Error

Software Common Cause Risk Mitigation Strategies

- Dynamic phases are riskiest times for software common cause due to short time-to-criticality
  - Ascent, Rendezvous, Entry
  - Mitigation strategies should be considered in relation to time-to-criticality, time-to-effect, time delay
  - During quiescent times with greater time-to-effect, there may be additional mitigation strategies available (i.e. software upload, reboot)
- Common mitigations for software common cause failures
  - Backup software – Set of dissimilar software automatically or manually engaged
    - Backup software systems can reside in same or different processor, depending on the failure cause
  - Manual piloting or control, crew/ground intervention
  - Reduced-capability primary software, software subset or “safe mode”
  - Uplink of new software
  - System Reboot

NASA Requirements for Software Fault Tolerance

- NPR 8705.2C: HUMAN-RATING REQUIREMENTS FOR SPACE SYSTEMS
  - 3.2.3 The space system shall provide at least single failure tolerance to catastrophic events, with specific levels of failure tolerance and implementation (similar or dissimilar redundancy) derived via an integration of the design and safety analysis.
  - 3.2.4 The space system shall provide the failure tolerance capability in 3.2.3 without the use of emergency equipment and systems.
  - 3.2.7 The space system shall provide the capability to mitigate the hazardous behavior of critical software where the hazardous behavior would result in a catastrophic event.
  - 3.3.2 The crewed space system shall provide the capability for the crew to manually override higher level software control and automation (such as automated abort initiation, configuration change, and mode change) when the transition to manual control of the system will not cause a catastrophic event.
- NPR 7150.2D: NASA SOFTWARE ENGINEERING REQUIREMENTS
  - 3.7.3 If a project has safety-critical software or mission-critical software, the project manager shall implement the following items in the software: [SWE-134] …
    - No single software event or action is allowed to initiate an identified hazard. …
- Software Assurance Standards to Assure these Requirements are Met:
  - NPR 8739.8A: SOFTWARE ASSURANCE AND SOFTWARE SAFETY STANDARD
  - NASA-STD-8719.13B: SOFTWARE SAFETY STANDARD

Selected Redundancy Requirements in Commercial Crew Program

ISS Crew Transportation and Services Requirements Document (CCT-REQ-1130)

3.2.3.1 Failure Tolerance to Catastrophic Events

The CTS shall provide failure tolerance for the control of catastrophic hazards, with the specific level of failure tolerance (one or more) and implementation (the use of similar or dissimilar redundancy) derived from an analysis of hazards, failure modes, and risk associated with the system.

a. The CTS shall provide dual failure tolerance or single failure tolerance with dissimilar redundancy for the control of catastrophic hazards in the systems that provide the guidance, navigation, and flight path/trajectory control functions for the deorbit burn, entry, and landing phases of the mission.

3.5.2.1 Emergency Entry Capability

The CTS shall provide emergency systems for entry, descent, and landing (EDL) to return the crew to Earth. [R.CTS.096]

3.8.4.1 Manual Control of Vehicle Flight Path

The spacecraft shall provide the capability for the crew to manually control the vehicle flight path, attitude, and attitude rates during all actively controlled phases of flight following the separation of the spacecraft from the launch vehicle, excluding ISS mated operations. [R.CTS.128]

Dissimilar Software Standard in Aircraft

- **From DO-178C:**

2.4.2 Multiple-Version Dissimilar Software

Multiple-version dissimilar software is a system design technique that involves producing two or more components of software that provide the same function in a way that may avoid some sources of common errors between the components. Multiple-version dissimilar software is also referred to as multi-version software, multi-version independent software, dissimilar software, N-version programming, or software diversity.

Software life cycle processes completed or activated before dissimilarity is introduced into a development, remain potential error sources. System requirements may specify a hardware configuration that provides for the execution of multiple-version dissimilar software.

The degree of dissimilarity, and hence the degree of protection, is not usually measurable. Probability of loss of system function will increase to the extent that the safety monitoring associated with dissimilar software versions detects actual errors using comparator differences greater than threshold limits. Dissimilar software versions are usually used, therefore, as a means of providing additional protection after the software verification process objectives for the software level, as described in section 6, have been satisfied.

Dissimilar software verification methods may be reduced from those used to verify single version software if it can be shown that the resulting potential loss of system function is acceptable as determined by the system safety assessment process.

Verification of multiple-version dissimilar software is discussed in section 12.3.2.

Assumes robust software development processes have already been followed, but uses dissimilarity for additional protection

Historical Analysis

- Historic Software Failure Incidents -
  - The team searched & studied incidents of software failures primarily at NASA or within aerospace
    - Total of 42 incidents were studied to characterize and determine software root cause
      - 24 open source – results shown here
        - » Aerospace Incidents (20) – loss of life, loss of mission, significant close-call
          - Significant Incidents & Close Calls in Human Spaceflight (Mercury, Gemini, Apollo, STS, ISS, Vashkod, MIR, EVA, Skylab, Ground facilities/chambers, https://sma.nasa.gov/SignificantIncidents)
        - » Medical (3) – loss of life/injury
        - » Commercial (1) – loss of service
      - 18 Proprietary aerospace
    - The team analyzed and characterized software root cause to determine -
      - Which is more prevalent – fail silent or erroneous?
      - What was the root cause of software failures?
      - Could the failures be corrected by reboot?
- Historic Mitigations (some examples follow)
  - Previous/current spacecraft were studied to understand their erroneous output mitigation strategies and "trip factors" for engaging backup systems
- Working on a paper to capture this preliminary analysis, sharing with community for feedback and examples of any more incidents (!)

Historical Software Incidents (1962-1969)

Year	Flight	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
1962	Mariner 1 Mission – Atlas-Agena Rocket	Programmer error (missing hyphen) in ground guidance veered launch vehicle off course	Mariner 1 was launched by an Atlas-Agena rocket from Cape Canaveral's Pad 12 on 22 July 1962. Shortly after liftoff, errors in communication between the rocket and its ground-based guidance systems caused the rocket to veer off course, and it had to be destroyed by range safety. The errors were traced to the omission of a hyphen-shaped symbol from one of the guidance program characters.	Loss of vehicle	Erroneous Output	Coding/Logic Error	No
1965	Gemini 3	Earth rotation omitted from landing location; crew manually corrected using lift to increase downrange.	During the first manned entry on March 23, 1965, the ground planning software omitted including the rotation rate of the Earth in calculating the splashdown location of the flight. The capsule would splashdown 190 miles up range of the intended target, far from the US Navy recovery ship.	Landed 60 miles short, due to crew ability to control the pitch of the capsule to maximize lift and downrange landing.	Erroneous Output	Coding/Logic Error	No
1965	Gemini 4	Erroneous entry data uplinked; crew manually corrected entry flight profile.	On June 7, 1965 the computer could not be updated for entry, could not be turned off, and then stopped working entirely. The crew resorted to a rolling Mercury-type entry, rather than the lifting bank angle the computer was supposed to help them achieve.	A non-lifting high-G Mercury-type ballistic entry was used. Landed 40 miles uprange of the target.	Erroneous Output	Data Misconfigured	No
1965	Gemini 5	Data error of earth rotation lands Gemini 5 short	Although the computer was operating, a programmer had entered the rate of the Earth's rotation as 360° per 24 hours instead of 360.98°. The crew compensated for the computing error, landing 80 miles (130 kilometers) short of the planned landing point in the Atlantic Ocean. Retrofire was initiated over Hawaii at 190 hours, 27 minutes, and 43 seconds into the mission. The astronauts controlled the reentry, creating drag and lift by rotating the capsule.	Landed 80 miles short	Erroneous Output	Data Misconfigured	No
1969	Apollo 10	Switch Misconfiguration	The Abort Guidance System (AGS) was inadvertently switched from HOLD ATTITUDE to AUTO, which caused the LM to look for the Command/Service Module (CSM) and flip end over end. The attitude indicator was going to the red zone and in danger of tumbling the inertial platform.	The Commander was able to grab the hand controller, switch to manual control, jettison the Descent Stage, control the LM Ascent Stage, and finally dock with the CSM.	Erroneous Output	Data Misconfigured	No

Historical Software Incidents (1981-1991)

Year	Flight	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
1981	**STS-1**	Failure of computers to sync to backup computer	OS when asked to initiate based on start time in the past will "slip" the start number of cycles needed to put it in the future. Seen as noise by backup computer. Placed all computers out of sync.	Backup computer wouldn't initialize	Fail Silent		No
1985-87	**Therac-25**	Radiation Therapy machine output lethal doses	Six accidents between 1985 and 1987, in which patients were given massive overdoses of radiation.: Because of several concurrent programming errors/race conditions some based upon the speed of operator input (slow operator input vs. fast input used in testing), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury. These accidents highlighted the dangers of software control of safety-critical systems and have become a standard academic case study. Additionally, the overconfidence of the engineers and lack of proper due diligence to resolve reported software bugs are highlighted as an extreme case where the engineers' overconfidence in their initial work and failure to believe the end users' claims caused drastic repercussions.	Four deaths, two chronic injured	Erroneous Output	Coding/Logic Error	No
1988	**Phobos-1**	Missing hyphen in unchecked uplinked command lost vehicle	On 2 September 1988, an expected transmission from Phobos 1 was not received. This was traced to a faulty key-command that was sent on 28 August from ground control in Yevpatoria. A technician unintentionally left out a single hyphen in one of the keyed commands. All commands were supposed to be proofread by a computer before being transmitted, but the computer that checked code was malfunctioning. The technician violated procedure and transmitted the command before the computer could be fixed to proofread it.[9] This minor alteration in code deactivated the attitude thrusters. By losing its lock on the Sun, the spacecraft could no longer properly orient its solar arrays, thus depleting its batteries.[10][11]	Loss of vehicle/Mission	Erroneous Output	Operator/Procedure Error	No
1991	**Patriot Missile**	Patriot missile failed intercept	1970s 24-bit legacy code rounding error in time conversion shifted missiles range gate. Error grew larger the longer software was run.	Failed to intercept Scud.	Erroneous Output	Coding/Logic Error	Yes

Historical Software Incidents (1994-2001)

Year	Flight or System	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
1994	**Pegasus XL STEP-1**	Booster loss of control due to lateral instability	Error caused by (1) gross underestimate of aerodynamic dihedral effect of high-wing, and (2) insufficient testing of faulty sideslip estimation algorithm that neglected gravitational acceleration	Loss of vehicle/Mission	Erroneous Output	Coding/Logic Error	No
1996	**Ariane 5 failure**	Ariane destroyed due to 64 bits of information stored into a 16 bit integer	Velocity error seen as new code with 64-bit floating point variable tried to put this into Ariane 4 legacy code 16-bit integer variable. New vehicle had higher velocity; data was diagnostic on previous vehicle but used as actual in Ariane 5.	Missile destroyed by maneuver to correct drift outside of missiles physical capability.	Erroneous Output	Coding/Logic Error	No
1999	**Mars Climate Orbiter**	Metric vs. imperial units error	Mars Climate Orbiter crashed in September 1999 because of a "silly mistake": wrong units in a program. The spacecraft encountered Mars on a trajectory that brought it too close to the planet, and it was either destroyed in the atmosphere or escaped the planet's vicinity and entered an orbit around the Sun.[2] An investigation attributed the failure to a measurement mismatch between two software systems: metric units by NASA and US Customary (imperial or "English") units by spacecraft builder Lockheed Martin.[3]	Loss of vehicle/mission	Erroneous Output	Data Misconfigured	No
2001	**Pegasus XL/HyperX Launch Vehicle / X-43A**	Airframe failure due to lateral instability leading to overstress of vertical fins	Error caused by (1) unfortunate combination of misestimates of aerodynamic characteristics and (2) aliased solid motor organ tone appearing as significant lateral acceleration at much low frequency, due to improper signal filtering	Loss of vehicle/mission	Erroneous Output	Coding/Logic Error	No

Historical Software Incidents (2001-2007)

Year	Flight or System	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
2001	**STS-108 through 110**	Shuttle main engine controller mix-ratio software coefficient sign-flip error	Prior to STS-108 a change had been made to the controller software coefficient for the Space Shuttle Main Engine (SSME) to compensate for an observed measurement bias in the SSME main combustion chamber pressure sensor, which controls the SSME fuel/oxidizer mixture ratio. The pressure chamber sensor was biased high causing the flight software to lower the chamber pressure by decreasing the liquid oxygen flow rate. Because of communication errors between ground systems engineers and deficiencies in the flight software verification and validation processes, the software coefficient was adjusted in the wrong direction, resulting in even larger dispersions in the mixture ratio and SSME performance.	SME underperformance close call -- though not extreme enough to not reach orbit or lose the mission.	Erroneous Output	Data Misconfigured	No
2003	**Multidata Systems Radiation Machine**	Radiation Therapy machine output lethal doses	National Cancer Institute, Panama City. In a series of accidents, therapy planning software created by Multidata Systems International, a U.S. firm, miscalculates the proper dosage of radiation for patients undergoing radiation therapy. Different calculations (double the dosage) resulted from the operator drawing the treatment region clockwise or counterclockwise.	15 deaths, many injured	Erroneous Output	Coding/Logic Error	No
2003	**Soyuz - TMA-1**	Ballistic reentry automatically engaged	Don Petit/Ken Bowersox ISS return flight. Automatic system commanded emergency ballistic mode as a backup. Radio antennae burned off so contact only established with crew on ground via hand-held radios [1]. Could not duplicate on ground but believed to be very small timing issue. [2]	460 km short [2].	Erroneous Output	Coding/Logic Error	No
2007	**F22**	F22 International Date Line	Multiple software-related systems failures when crossing the 180th meridian; failures resulted in loss of navigation and communication; clear weather allowed squadron to follow tankers back to Hawaii. Fail Silent - Backup Manual Systems Employed	Loss of navigation & communication, relied on other aircraft to compensate for lost functionality.	Fail Silent		No

Historical Software Incidents (2008-2012)

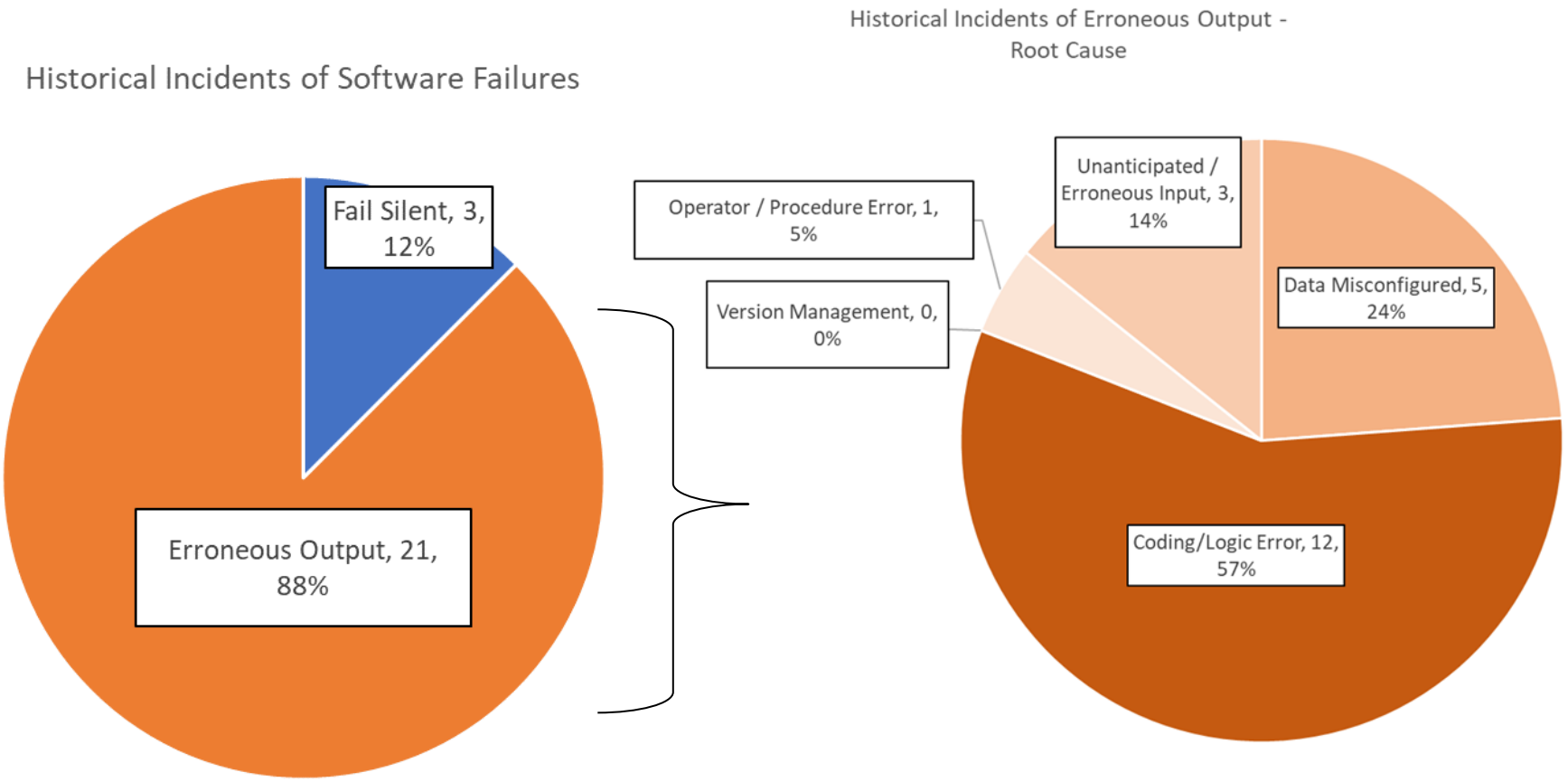
Year	Flight or System	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
2008	**STS-124**	All 4 shuttle computers fail during fueling	A 1-1-1-1 system disagreement among 4 redundant computers due to a cracked diode in another box that effectively sent each computer a different signal with the same sensor input.	All 4 computers showed disagreement, shut down fuel attempt	Erroneous Output	Erroneous Input	No
2008	**Quantas Flight 72, Airbus A330-303**	unanticipated input spikes caused autopilot to pitch-down resulting in passenger injury	Air Data Inertial Reference Units gave "spiked" data to autopilot. Design never considered "spiked" data. Systems warning irregularity, including contradictory stall and overspeed warnings followed by uncommanded pitch down. The Australian Transport Safety Bureau (ATSB) investigation found with a previously unknown software design limitation of the Airbus A330's fly-by-wire flight control primary computer (FCPC).	one crew member and 11 passengers suffered serious injuries	Erroneous Output	Unanticipated Erroneous Input	No
2008	**B-2 Spirit - Guam crash**	Miscalculation in flight computers with missing input data calculated uncommanded pitch up.	Because three pressure transducers failed to function (attributable to condensation inside devices and heavy rain) the flight-control computers calculated inaccurate aircraft angle of attack and airspeed. Higher indicated speed than actual, and once airborne gave a negative angle of attack (AoA), causing an uncommanded pitchup.	Crew members successfully ejected.	Erroneous Output	Unanticipated Erroneous Input	No
2012	**Red Wings Flight 9268 TU-204 crash**	Unanticipated landing circumstances coupled with design features resulted in crash landing	Crosswind "light" landing in rain, weight on wheels switch failed to engage, aircraft hydroplaned, reverse thruster did not deploy. Erroneous input not recognized by crew. Pilots, unaware reverse thrusters didn't deploy. As a safety feature, both sets of main landing gear were required to be compressed simultaneously before the thrust reversers could deploy. Because there was no compression of the right landing gear, the reversers were never deployed, and moving the controls to the Maximum Reverse position caused an increase of forward thrust in both engines. In addition to the lack of reverse thrust, the airbrakes and spoilers failed to activate automatically, and the crew did not attempt to activate them manually.	5 of 8 crewmembers killed	Erroneous Output	Coding/Logic Error	No

Historical Software Incidents (2018-2020)

Year	Flight or System	Title	Cause/Description	Result / Outcome	Erroneous or Fail Silent?	If Erroneous - Type	Recoverable with Reboot?
2018, 2019	**737 Max crash**	Unanticipated / untrained software response to faulty sensor input	Faulty AoA sensor showed higher than actual AoA. System to counter weight and balance concerns would trim pitchdown to counter. Note that community was not informed of this software change, nor were many (especially non-Western) pilots trained on how to respond. Erroneous input not in design, no crew training on auto response	Trim automatically went to pitchdown to counter erroneous AoA.	Erroneous Output	Unanticipated Erroneous Input	No
2020	**Amazon Web Service (AWS) Kinesis**	Loss of service	OS upgrade had a limit set on number of threads, which was exceeded when Amazon tried to scale up service. The failure cascaded from server to server which shut down entire system. Excellent software development processes. Fail Silent - loss of Service, coding/logic error in OS.	Server failure shed load to another server, where it failed too, cascading to outage.	Fail Silent	Coding/Logic Error	Yes
2020	**BD Alaris™ Infusion Pump**	Infusion delivery system faults cause injury/death	Software errors could lead to over/under infusion, infusion delay, infusion interruption. When programming, if more than one function is selected within a one-second interval, it "results in a non-silenceable, high priority alarm and status indicator lights on modules will flash red."	55 injuries, 1 death; Class I FDA recall issued	Erroneous Output	Coding/Logic Error	No

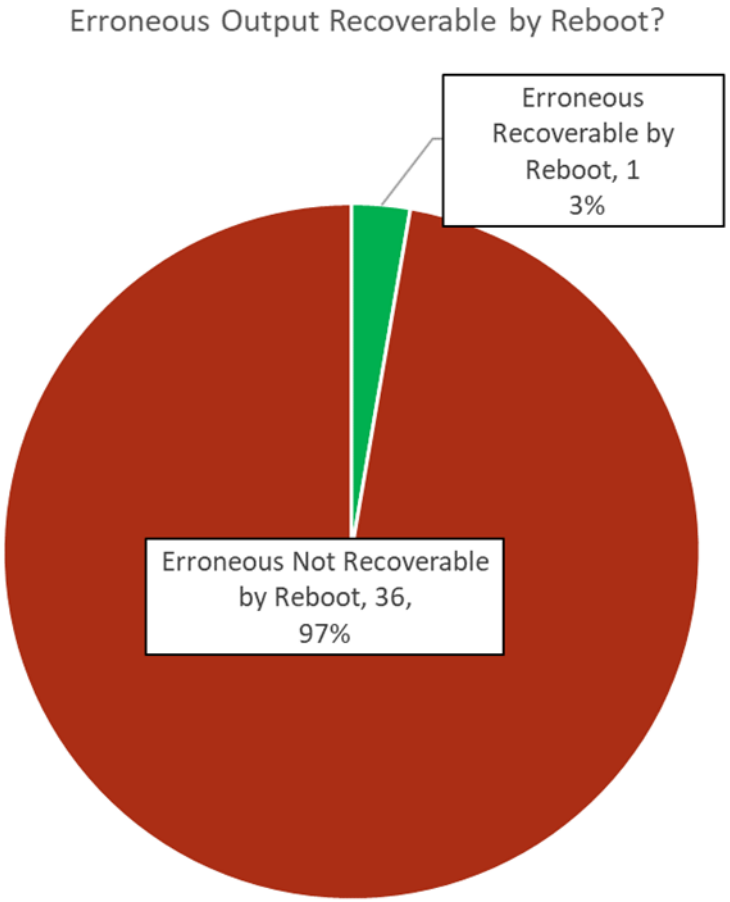
Software Failure Statistics - Categories

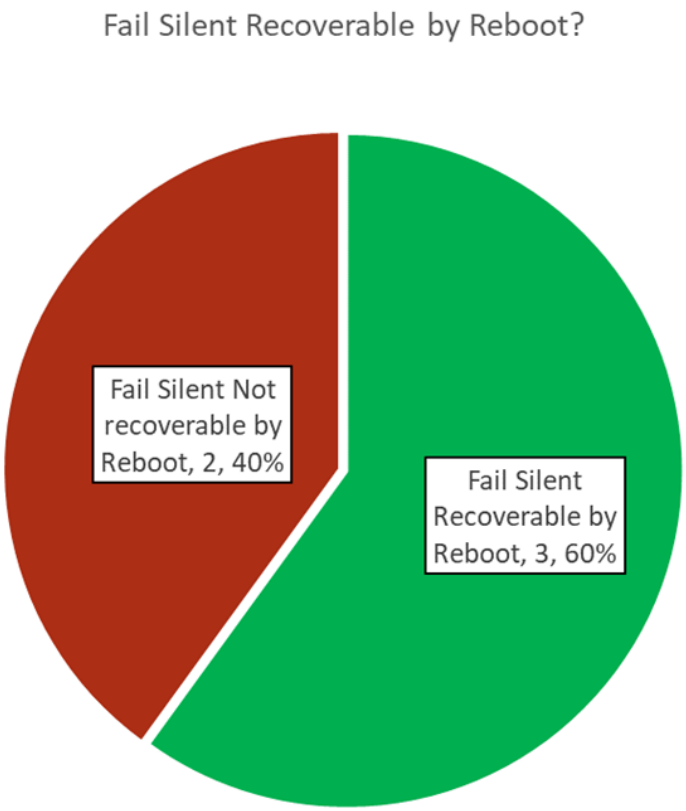
- A NASA Engineering and Safety Center (NESC) Study [TI-22-01725] of 24 incidents - 20 aerospace software failure incidents, 3 medical, 1 commercial categorized the failures as follows



- ***Takeaways:***
  - ***Erroneous Output Situations were over 7x more historically prevalent than fail silent cases (88%-12%)***
  - ***Coding/logic (includes requirements) and Data Configuration errors were the root cause of most erroneous output situations***

Software Failure Statistics – Reboot Recoverability





- ***Takeaways:***
  - ***Rebooting is predominantly ineffective to clear/recover from erroneous output situations***
  - ***Not all fail-silent cases are recoverable by reboot – those errors that grow over time seem recoverable***

Software Common Cause Failure Summary

- Software “common cause” or “common mode” failures are instances when a single software error, even if the same software load is replicated in multiple machines, can result in unexpected vehicle/space system behavior
- Software in Space Systems should be architected for redundancy based on criticality and time-to-effect, with requirements driven primarily by NPR 8705.2C and NPR 7150.2D
  - There are several common mitigation strategies to achieve fault tolerance, including dissimilar backup or manual control systems
- Software Errors can be described in two classes:
  - Fail silent – software stops outputting (less common)
    - Fail silent causes include operating system halt, memory access violations, priority inversion
    - Some of these can be cleared by reboot
  - Fail Erroneously – software does the wrong thing (more common)
    - Erroneous output manifestations are much more common (7x) than “fail silent”
    - Erroneous Output Causes include Data Misconfiguration, coding/logic errors, improper versioning, unanticipated input, and procedural operator error
    - Erroneous Output situations are not likely to be cleared by reboot