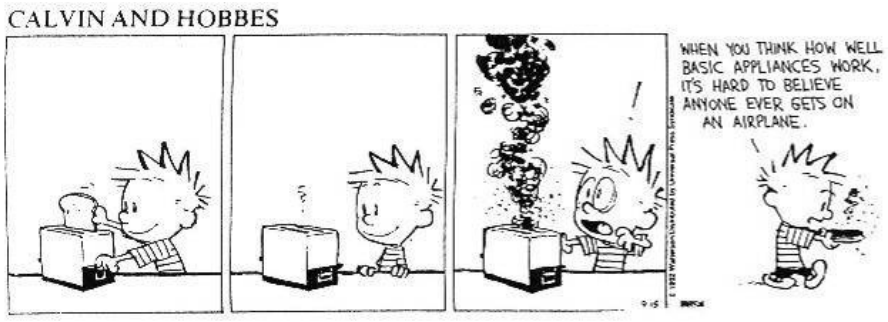
Abstract - Designers of safety-critical medical actuator control applications face rigorous requirements and standards to assess safety requirements, develop system architectures, and design component hardware and software. This paper demonstrates integrated techniques of safety-critical development with examples from various actuator applications. Strategies for safety analysis, engineering design and application of lifecycle guidelines are discussed. Methods of developing actuator controls with robust fault tolerance and testability are highlighted. The secrets of effective safety-critical development are revealed.

Stupid is as stupid does. -- Forrest Gump



Eventually, everyone screws up and everything breaks.
If your world is void of failures, you don't need to be here.
But if safety-related failures worry you, stick around.

Speaker Background – VPS, OTTD, we do PD&M, R&D TS,

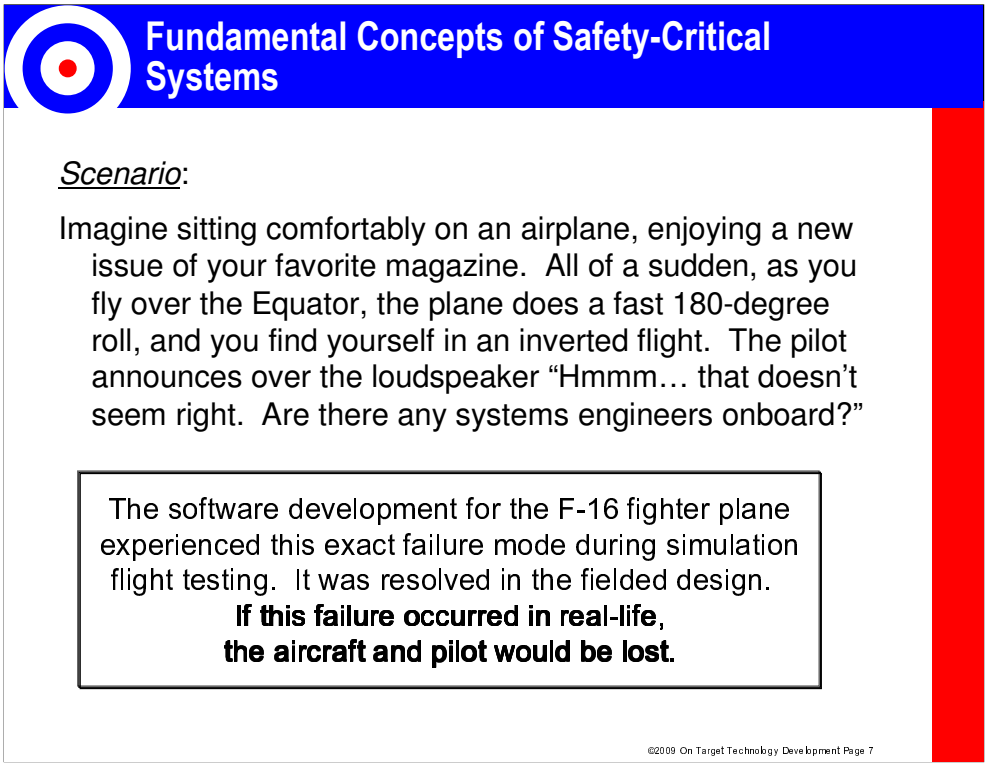
Vincent Socci is a cross-disciplined systems, hardware and software engineer. His technology expertise includes embedded systems, sensors and signal processing, power control systems, and diagnostics. Socci has 20 years of experience in safety-critical systems development. He holds an MBA in technology management, and MS and BS degrees in electrical engineering. As Principal of On Target Technology Development, Socci supports clients with electronics design and embedded controls development. He has applied the safety-critical design concepts presented in this paper in medical, aerospace, automotive, locomotive, and industrial applications.

Agenda

- Fundamentals of Safety-Critical Systems
- Actuator Control Design from a Systems Perspective
- Industry Specifications and Standards
- Development Strategies
- Example Cases
- Conclusions
- Questions and Answers


Industrial Trends in Safety-Critical Systems
Medical
Growing focus on safety development standards
Sensitive to failure, development cost/time, marketability, global competition
Aerospace
Long history of fly-by-wire and other x-by-wire actuator control systems
Long development, low volume, expensive safety-critical applications.
Automotive
Emerging x-by-wire needs
Short development, high volume, cost-efficient safety-critical applications.
Demand for fault tolerance, speed-to-market & low cost.
©2009 On Target Technology Development Page 5


Fundamental Concepts of
Safety-Critical Systems
©2009 On Target Technology Development Page 6



Sound far fetched?

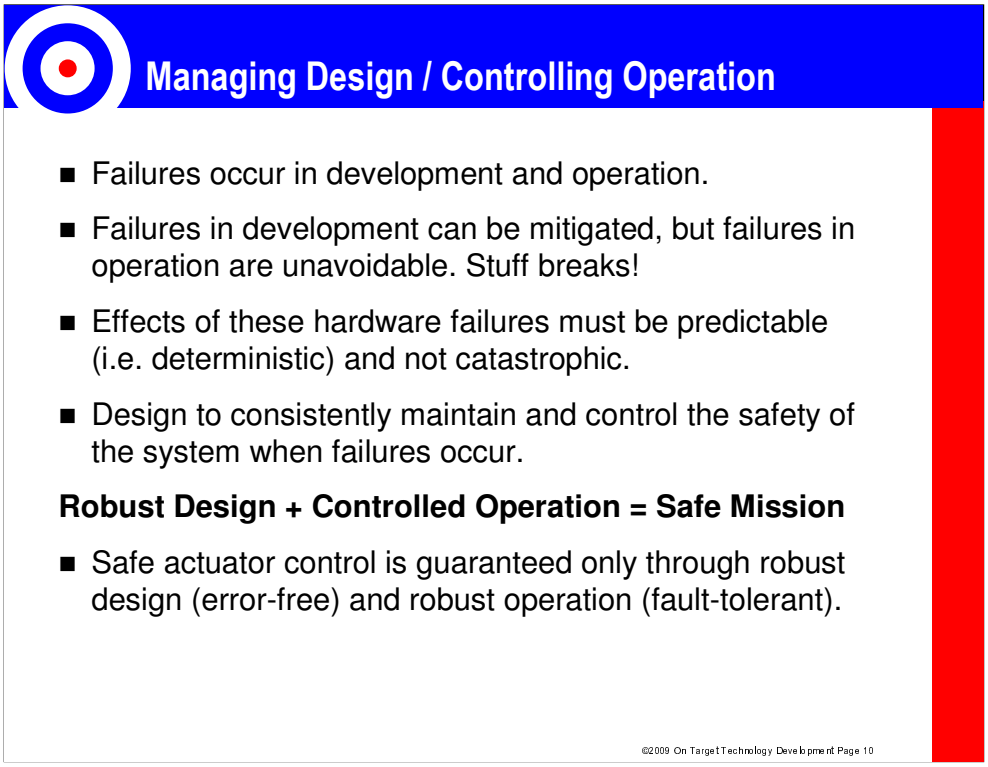
Other F-16 Simulation Failures

- When the computer was commanded to raise the landing gear while the aircraft was standing still on the runway, the computer complied and “turfed” the aircraft.
- The aircraft system also complied with commands to jettison missiles, bombs, fuel tanks, etc. while the plane was upside-down, resulting in them falling on and damaging the wings.
- When the F-16 went into a spin, the software did not give the pilot enough control authority to recover. The pilot had to eject.

Failures like this put lives and property at risk.
We can't let them happen – period.

Safety-related vs. Safety-Critical

- Safety-related systems – those in which a failure during operation can have serious or irreversible effects – such as loss of life or limb, severe property damage or financial losses. (e.g. insulin pump)
- Safety-critical systems – safety-related systems that present a direct threat to human life. They can include aircraft control systems, medical instrumentation, railway signaling, nuclear reactor control systems and many other applications.
- Safety-critical systems include all of the components that work together to achieve the safety-critical mission.
  - These may include input sensors, digital data devices, hardware, peripherals, drivers, actuators, the controlling software, and other interfaces.
  - Their development requires rigorous analysis and comprehensive design and test.



Safety-related failures can occur in development or in operation. The design may be weak or not robust enough for the operating environment. Design standards and practices, including rigorous requirements management, engineering analyses, design reviews and testing can support validation of the design.

Failures in operation are unavoidable. Stuff breaks! Whether it's a wire in a harness or a worn-out relay – hardware will eventually fail. Safety-critical systems are designed such that the effects of these hardware failures are predictable (i.e. deterministic) and not catastrophic. Architectures are designed to consistently maintain and control the safety of the system when failures occur.

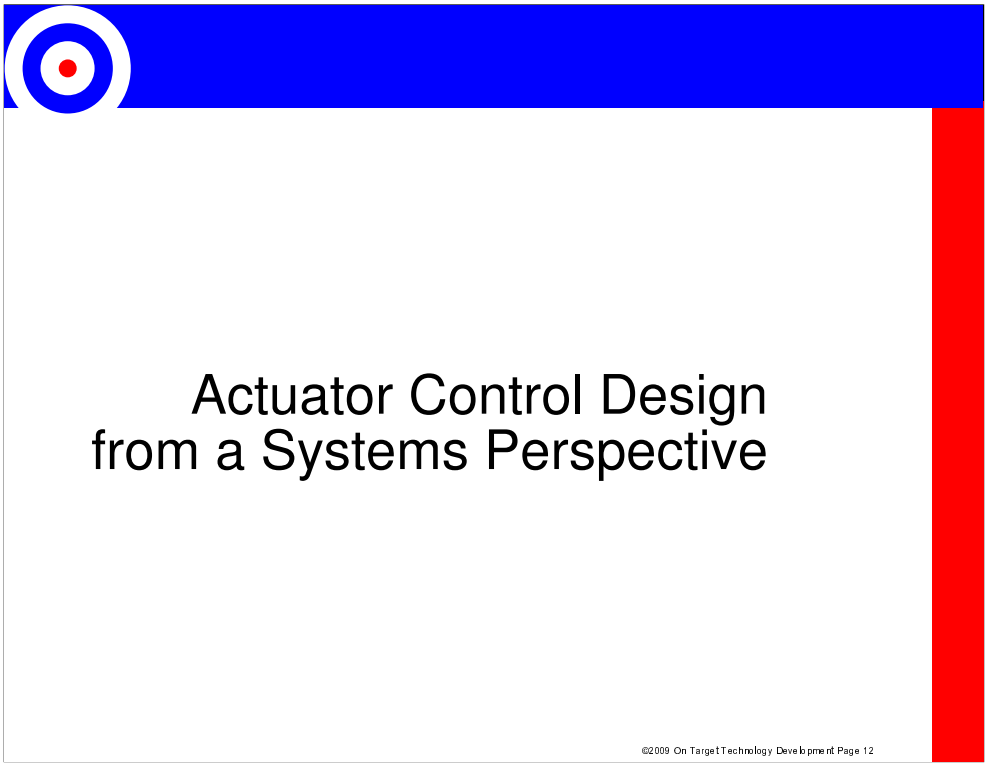
Robust Design + Controlled Operation = Safe Mission

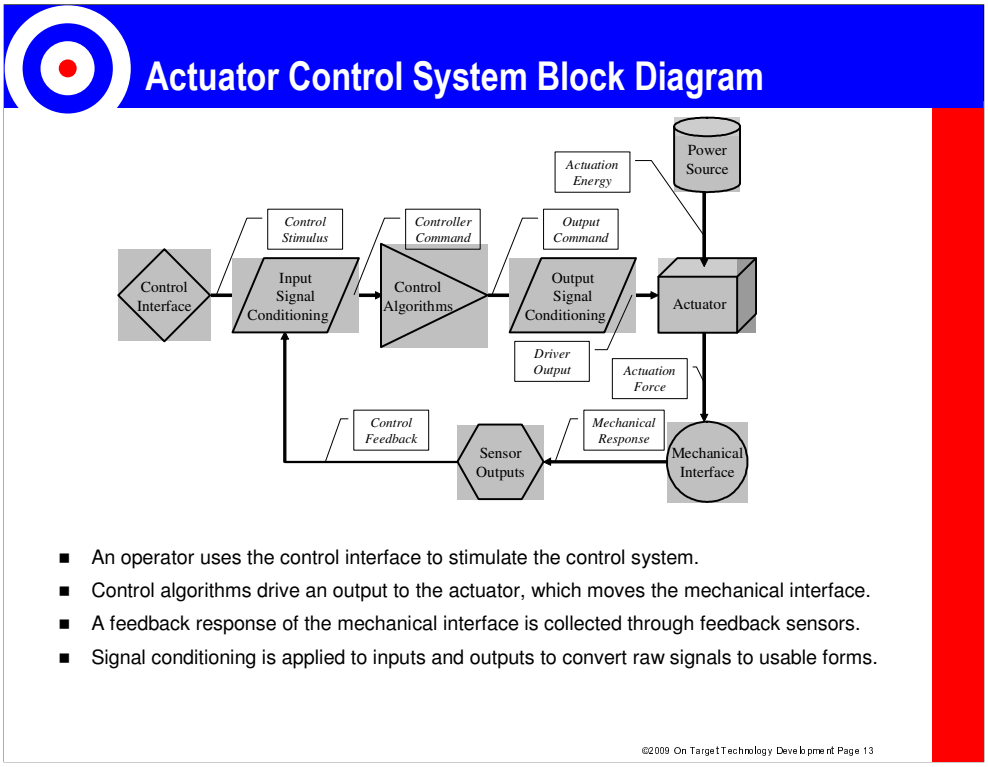
Remember that safe actuator control is guaranteed only through robust design and robust operation. Design robustness is achieved through robust processes and engineering design practices. Controlled operation is achieved by thorough monitoring, fault detection and mitigation strategies.

Safety-Critical Medical Actuator Examples

- Hospital beds
- Ambulatory scooters
- Patient lifts
- Pumps and valves for drug delivery
- X-ray controllers
- Massage therapy
- Prosthetics
- Laparoscopic tools
- Etc.

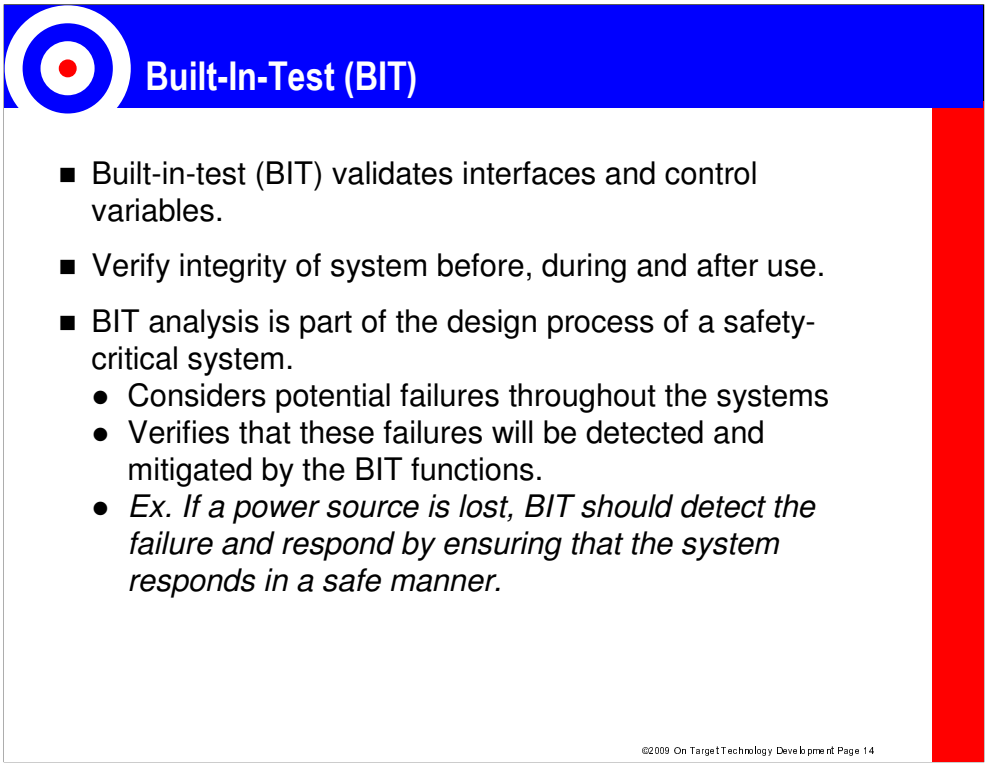
Monitors are passive, but actuators are active.


Actuator Control Design
from a Systems Perspective
©2009 On Target Technology Development Page 12



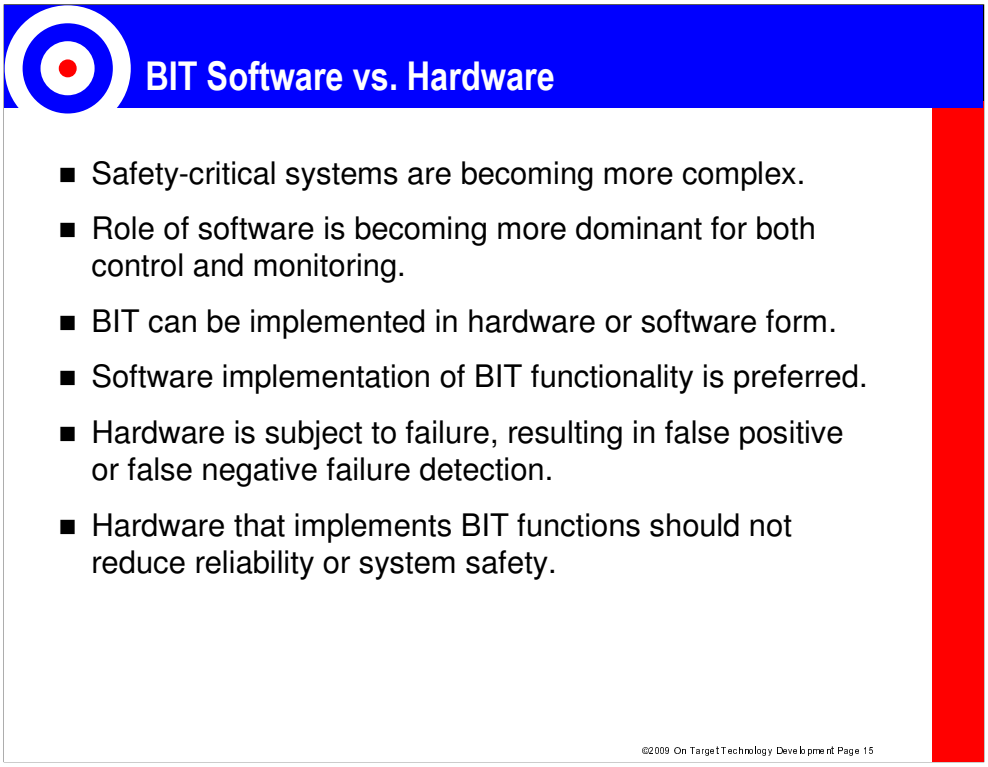
This paper focuses on actuator control systems because these applications provide a relevant mixture of hardware and software functions that integrate the broad requirements of safety-critical systems. Consider the following system block diagram of an actuator control system.

Most of these control system functional blocks are hardware functions, implemented by some mechanical or electrical components. The control algorithms are typically software functions, implemented by a computing program installed to and operating on a processing device. Signal conditioning can occur in both hardware and software.

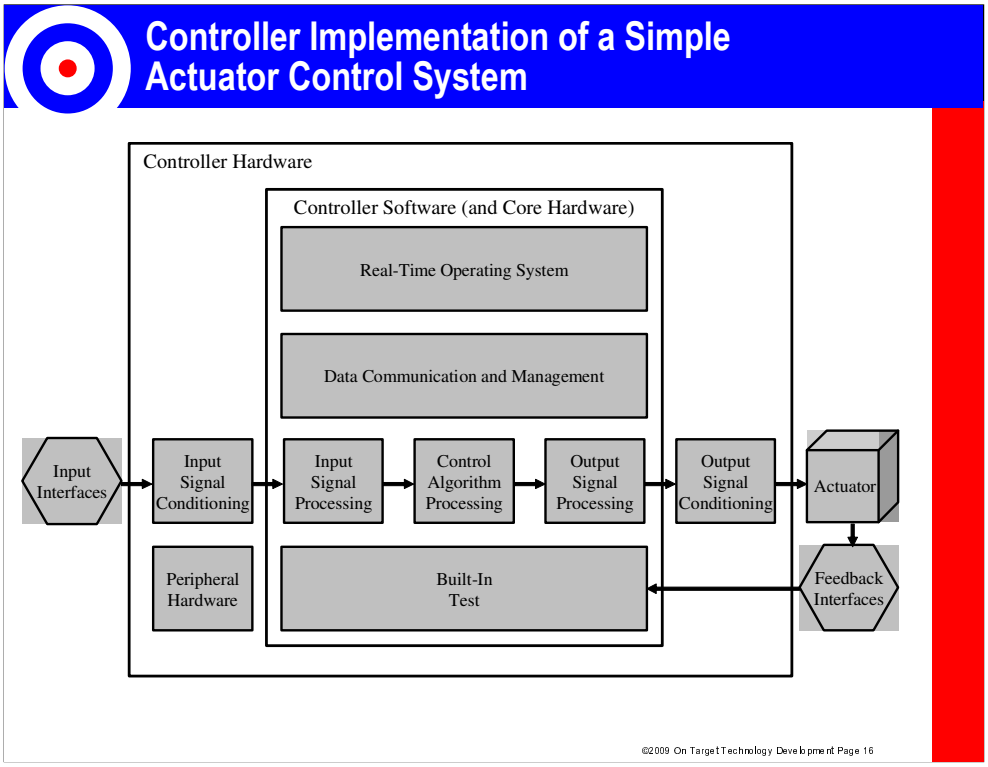


Controllers generally use some form of built-in-test (BIT) to validate interfaces and control variables. BIT is designed to detect failures in any of the system elements. BIT is the primary means to verify the integrity and operation of system and hardware components before, during and after the mission.

Systems engineers typically perform a BIT analysis as part of the design process of a safety-critical system. This BIT analysis considers potential failures throughout the systems and verifies that these failures will be detected and mitigated by the BIT functions. For instance, if the power source of Figure 2 fails (e.g. loss of power), BIT should detect the failure and respond by ensuring that the system responds in a safe manner.



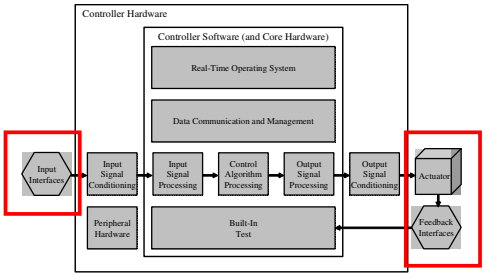
As safety-critical systems are becoming more complex and computer-controlled, the role of software is becoming more dominant for both control and monitoring. BIT functionality can be implemented in hardware or software form. Software implementation of BIT functionality is preferred. Hardware that implements BIT functionality can itself be subject to failure, resulting in false positive or false negative failure detection. Engineering design practices suggest that hardware that implements BIT functions should not reduce reliability or system safety.



Now let's consider a simple actuator system from the perspective of controller implementation. This figure provides a general block diagram of controller implementation. The arrows represent a simplified functional flow for the actuator. Actual data flow can be much more complex. Refer to the figure for the following discussion.

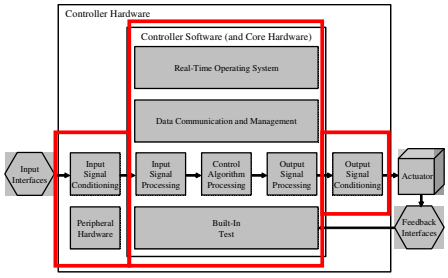
External Interfaces

- The following interfaces are external to the controller hardware:
  - Actuator
  - Input interfaces
  - Feedback interfaces.
- Near controller or far away
- Connected through wire harnesses, or wireless
- Susceptible to hardware failures



Controller Hardware

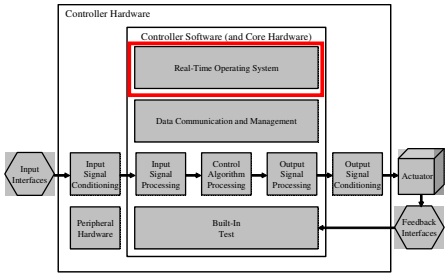
- Includes:
  - I/O signal conditioning
  - peripheral interfaces
  - core processing platform.
- Signal conditioning:
  - Converts the input signals for the core processor
  - Translates processor outputs into driver signals for the actuator.
- Peripheral hardware may include watchdog monitors, protective devices etc.
- Processing platform includes the processor, memory, and any other devices needed to support processing operation.



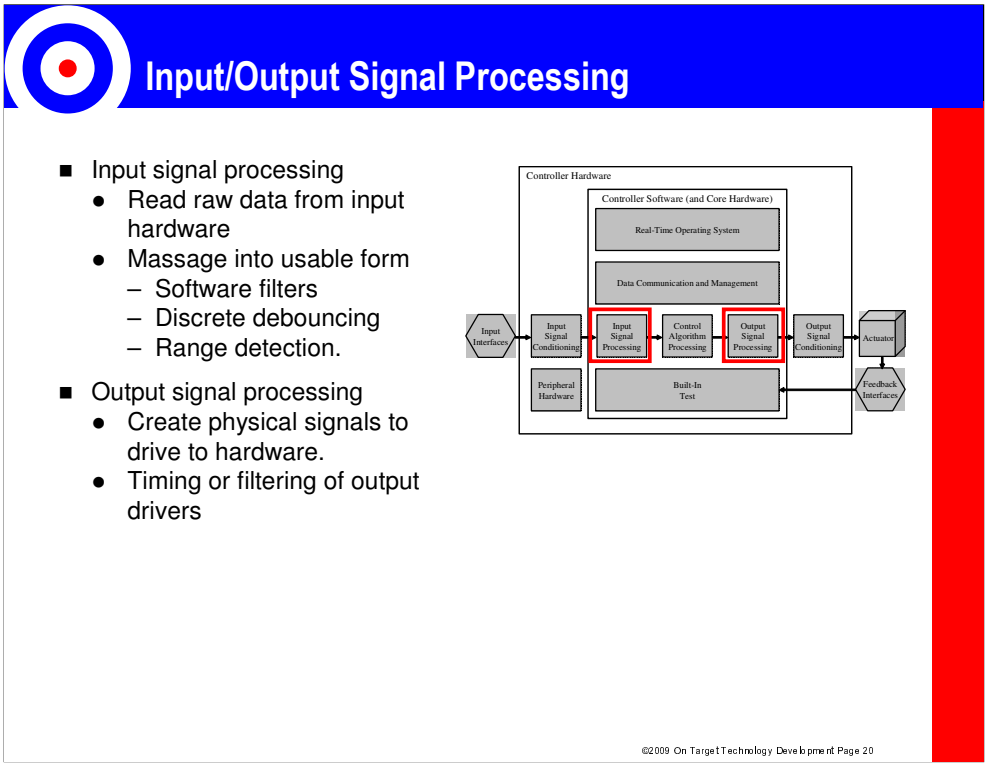
The controller hardware provides I/O signal conditioning, peripheral interfaces and a core processing platform. Signal conditioning converts the input signals into forms that can be interpreted by the core processor, and translates processor outputs into driver signals for the actuator. These driver signals may be high current signals that directly drive the actuator or they can provide indirect control of an external power source, as shown in Figure 2. Peripheral hardware may include watchdog monitors, data communication devices, power converters, protective devices and filters, configuration hardware and other peripheral hardware. The core processing platform includes the processor, memory, and any other devices needed to support processing operation.

Controller RTOS

- Real-time Operating Systems perform:
  - executive management
  - task scheduling
  - other operating utilities
- Examples:
  - Ad-hoc operating system
  - Commercial off-the-shelf RTOS:
    - Integrity, VxWorks, LynxOS
    - Must be certifiable to the safety criticality level of the application.



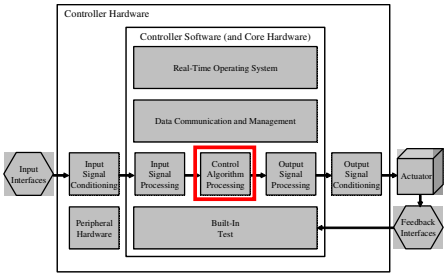
Controller software, which resides and operates on the core hardware, has a real-time operating system (RTOS) that performs all executive management and task scheduling, as well as other operating utilities. This may be an ad-hoc operating system, or it can be a commercial off-the-shelf RTOS, such as VxWorks, CsLEOS or Integrity, that is certifiable to the safety criticality level of the application.



Input signal processing software reads raw data inputs from the hardware and massages them into usable form for the software algorithms. This massaging includes software filters (e.g. lag and lead filters), discrete debouncing, and range detection. Output signal processing converts the results of the control algorithms into physical signals to drive to hardware.

Control Algorithm Processing

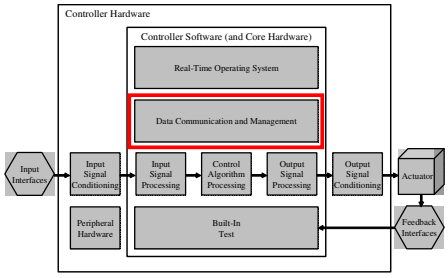
- Provides the heart of the actuator control functionality.
- Control data and variables are read and processed to ***deterministically*** drive the real-time response of the actuator.
- Critical real-time and frequency-dependent algorithms



Control algorithms provide the heart of the actuator control functionality. This is where the plant models, control loops and decisions are processed. Control data and variables are read and real-time algorithms are processed to deterministically drive the appropriate real-time response of the actuator.

Data Communications and Management

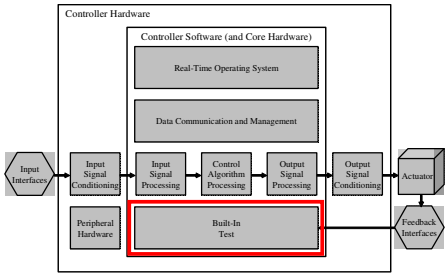
- Manipulate the data associated with the inputs and outputs
- Provides any processing associated with redundancy or cross-channel communication.
  - Multichannel Datasets (copies)
  - Data synchronization
  - Self and cross-channel BIT.
- Amount of data that must be managed can become huge and coordination can become complex.



The data communication and management functions manipulate the data associated with the inputs and outputs, including digital communication to external components. These functions also provide any processing associated with redundancy or cross-channel communication. For example, a quad-redundant actuator system would have a complete set of data for each of the four channels. Some level of data synchronization is needed to maintain coordinated control of the four channels. Each channel may perform BIT on itself and one or more of the other channels. Depending on the intricacy of data communication, the amount of data that must be managed can become huge and coordination can become complex.

Built-In-Test

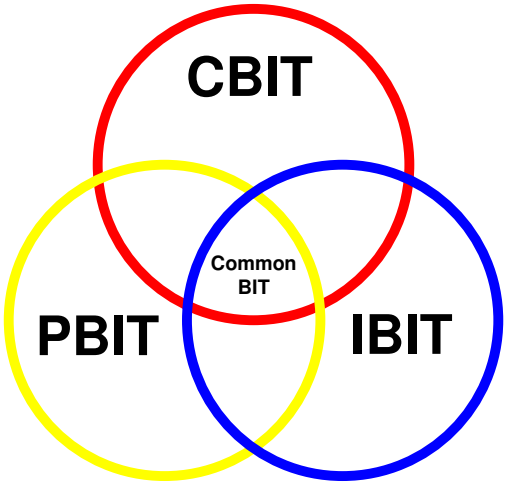
- BIT functions **detect** faults and **isolate** them to individual components or small groups.
- Detection Test during operation and compare results to expected values.
- Isolation Determine if the fault occurred inside the controller or external interfaces.
- Test all faults that have an effect on the safe operation of the mission.



Actuator controls include BIT functions to detect faults and isolate them to individual components or small groups of them. Failure detection is usually accomplished by conducting a series of tests during operation and comparing results to expected values. Isolation logic determines if the fault occurred inside the controller or was the result of external interfaces. The designer's goal is to test any potential fault that has an effect on the safe operation of the mission in the appropriate mode of BIT.

BIT Modes

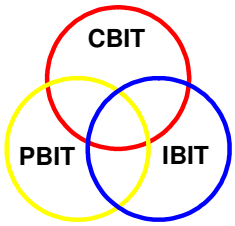
- BIT can be categorized into the following modes:
  - Power-up BIT (PBIT)
  - Continuous BIT (CBIT)
  - Initiated BIT (IBIT)
- Each safety-critical fault is tested in one or more of these BIT modes
- Each BIT mode may perform a subset (or superset) of another BIT mode
- Fault thresholds, persistence limits and scheduling of individual BIT are coordinated to reject false failure indications.



Nested acronyms

Power-up BIT (PBIT)

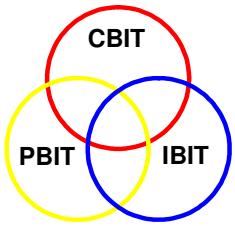
- Also known as Start-up BIT
- Executes on application of power
- Rapid check of ability to operate
- Examples
  - Core processing failure – the mission can be halted before operator safety is jeopardized
  - Actuator feedback interface failure – the actuator may not be controllable, and the mission would be aborted
- Scope of PBIT testing depends on application power up requirements.
- Designers must trade-off PBIT test coverage with PBIT timing constraints.



- Sample PBIT Tests:
  - Processor diagnostics
  - Memory
  - Configuration
  - Watchdog timeout
  - Power supply voltage
  - Interrupts
  - Critical I/O

Continuous BIT (CBIT)

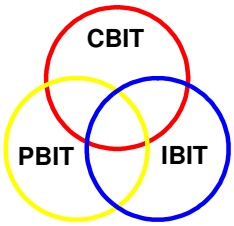
- Also known as Periodic BIT
- Provides continuous monitoring of all system components.
- Minimizes failure exposure time.
- Example: Current feedback on a drug pump actuator detects if the flow is obstructed during operation.
- CBIT completion time must be considered when the maximum failure exposure time (the maximum time between when a fault occurs and when the fault is detected and mitigated) is critical.



- Sample CBIT Tests:
  - Subset of PBIT tests
  - Control Sensors
  - Control Discretes
  - Feedback Sensors
  - Feedback Discretes
  - Dynamic responsiveness
  - Data communication validation
  - Actuator current monitors

Initiated BIT (IBIT)

- Also known as Maintenance BIT, or other alternative names.
- Extensive set of tests, initiated by the operator, which occurs when the system is in a stable, known environment.
- The environment must be controlled because control of the actuator is given to the BIT functions rather than the normal control algorithms.
- Example: range test on an actuator
- IBIT completion time is typically long because it performs comprehensive, full-range tests

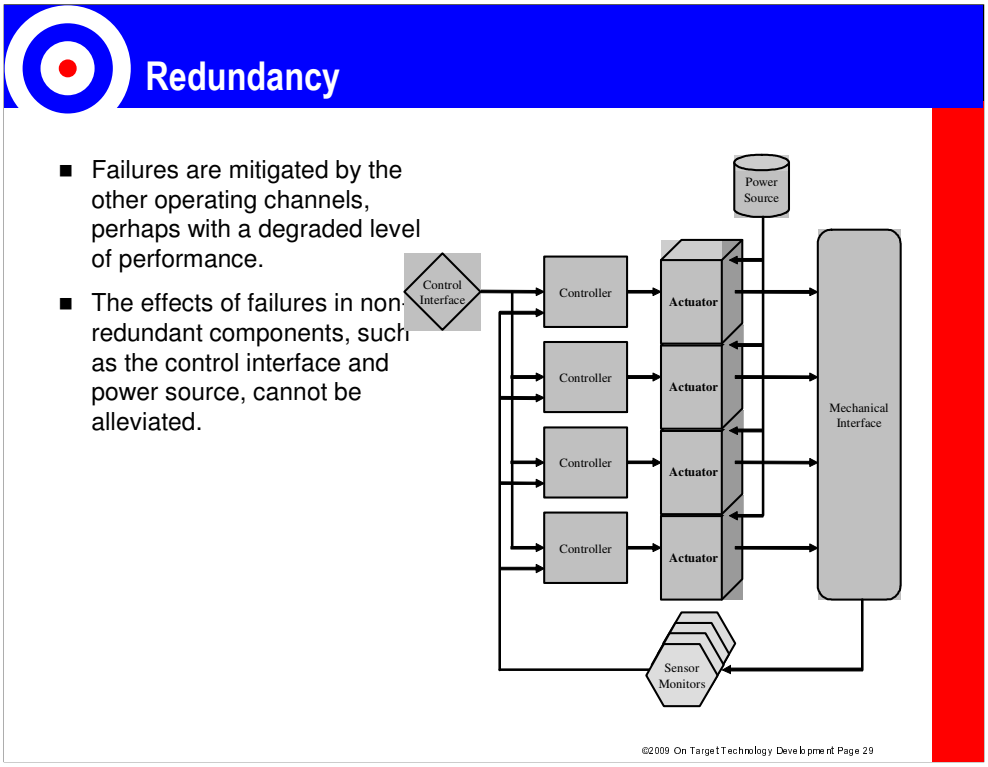


- Sample IBIT Tests:
  - Superset of PBIT and CBIT tests
  - Power system tests
  - Dynamic actuator tests
  - Mechanical range tests
  - Initiated failure tests

Suppose an actuator moves a swing-arm from 0º to 180º. The system cannot test the full range of motion during operation, unless the application guarantees to move the swing-arm over its full range of motion during the mission. The operator can use a special controlled test in IBIT to verify that the swing-arm can operate over its full range of motion.

BIT Analysis

- Leverages results of other system safety analyses, including:
  - Preliminary System Safety Assessment
  - Fault Tree Analysis
  - Failure Modes and Effects Analysis
- Consider failure mode, probability of failure, cause and effect, recognition/detection scheme, isolation, system compensation
- Will typically yield action items for design improvements.
- Not a “do it once checkbox” – Analysis is incremental until requirements are satisfied.

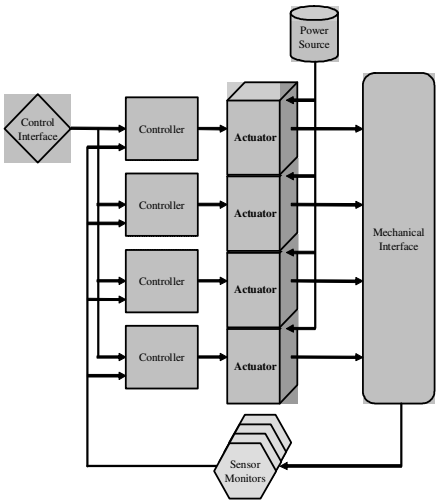


Redundancy is a common design practice that is often used to maintain functionality after a failure occurs.

In this case, functionality is maintained by four redundant operating channels. Failures that occur in a controller, actuator or mechanical monitor can be mitigated by the other operating channels, perhaps with a degraded level of performance. The effects of failures in non-redundant components, such as the control interface and power source in this case, cannot be alleviated.

Degradation Categories for Failure Response

- *Fail-Operational* – From an operator perspective, the system behaves normally.
  - Failure is reported, but system operation is unaffected.
  - System components that remain functional typically take over the functions of the failed components.
  - *Example*: Dual-redundant surgical blood flow systems typically can meet performance requirements with only one channel. When one channel fails, the system continues to perform using the remaining channel.

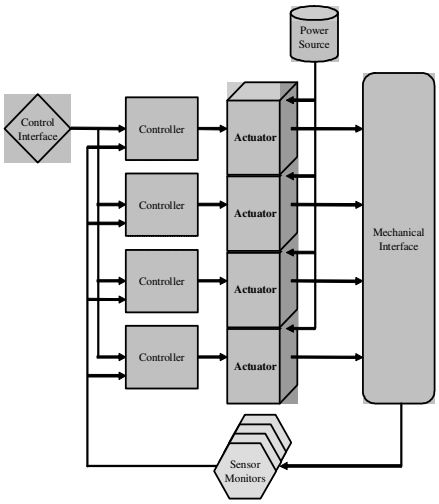


There are many failure modes, but system designers have developed approaches to mitigate them. Various response categories have been developed to describe mitigation of these system failures:

Fail-Operational – From an operator perspective, the system behaves normally. The failure is reported, but system operation is unaffected. The system components that remain functional typically take over the functions of the failed components. *Example*: Quad-redundant flight control surface actuator systems typically can meet performance requirements with only three channels. When one channel fails, the system continues to perform using the remaining three channels.

Degradation Categories for Failure Response

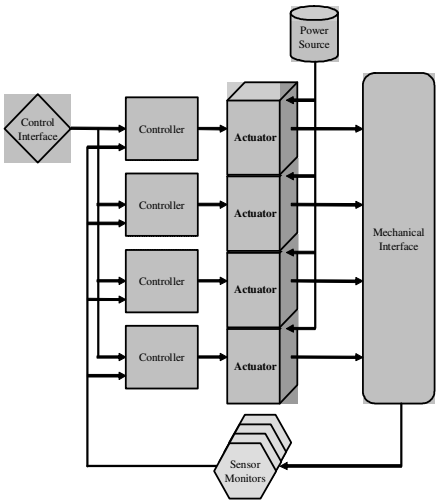
- *Fail-Passive* – Outputs assume a predetermined desirable state, such as a power disconnect to the actuators.
  - Failure is reported, and operation is reverted to backup.
  - Intervention may be required, but the system remains under control in some degraded fashion.
  - *Example*: Ambulatory scooter used reduced speed mode when battery supply is low. The vehicle will not maintain desired performance, but rider can "limp" home.



Fail-Passive – The system outputs a predetermined desirable state, such as a power disconnect to the actuators. The failure is reported, and operation is typically reverted to a backup mechanism. Intervention may be required, but the system generally remains under control, although usually in some degraded fashion. *Example*: Many marine actuator control applications provide a mechanical backup that is automatically engaged when automated systems fail, thereby enabling the pilot to "limp home". The vehicle may not meet performance requirements, but it will be able to travel to a safe environment.

Degradation Categories for Failure Response

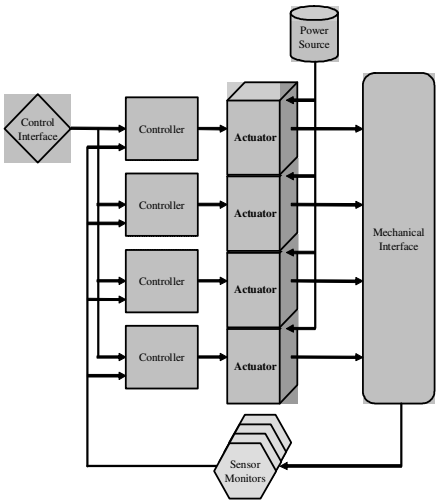
- *Fail-Safe* – Provides a safe response when normal, predictable control is not possible.
  - Subcategory of fail-passive
  - Actuators are not controllable in a manner to meet the performance specification.
  - System employs mechanisms to force the actuator in a known state that maintains a safe operational state.
  - *Example*: Patient lift backup is engaged when automated systems fail, holding patient safely. The lift will not meet performance requirements, but it will maintain a safe environment.



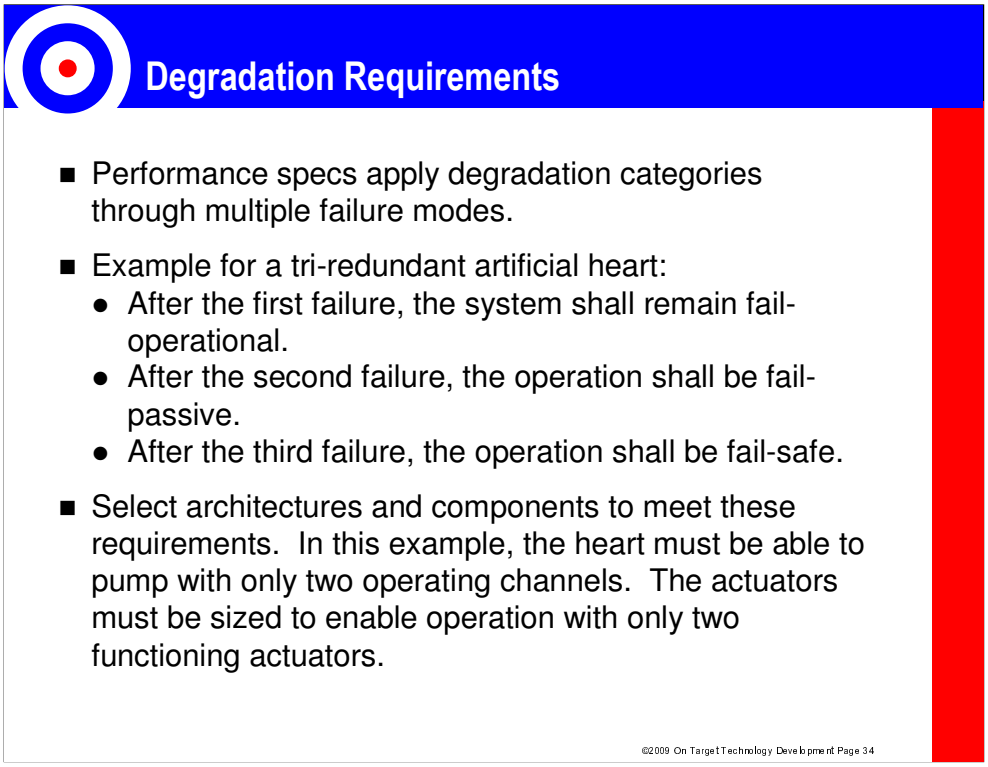
Fail-Safe – This approach is used to provide a safe response when normal, predictable control is not possible. In this category, which is sometimes considered a subcategory of fail-passive, the actuators are not controllable in a manner to meet the performance specification. However, the system employs mechanisms to force the actuator in a known state that maintains a safe operational state. *Example*: Hybrid-electric vehicles (HEV) have experienced failures that cause "runaway motors" where motors speed up out of control and transfer uncommanded power to the drive axles. A fail-safe mitigation would disconnect power from the motor, usually through high-current relays, when a failure occurs. The vehicle may not be drivable, but the passengers and cargo remain safe.

Degradation Categories for Failure Response

- *Fail-Active* – Highly undesirable state occurring when mitigation into one of the other categories is not possible.
  - Applications allow a small probability, such as 10^{-9}.
  - Actuators respond in an uncontrollable/unpredictable (nondeterministic) manner.
  - Safety-critical systems are designed to minimize fail-active scenarios.
  - *Example*: Automotive steer-by-wire applications have steering wheel interfaces with common mode failures that could inhibit steering control.

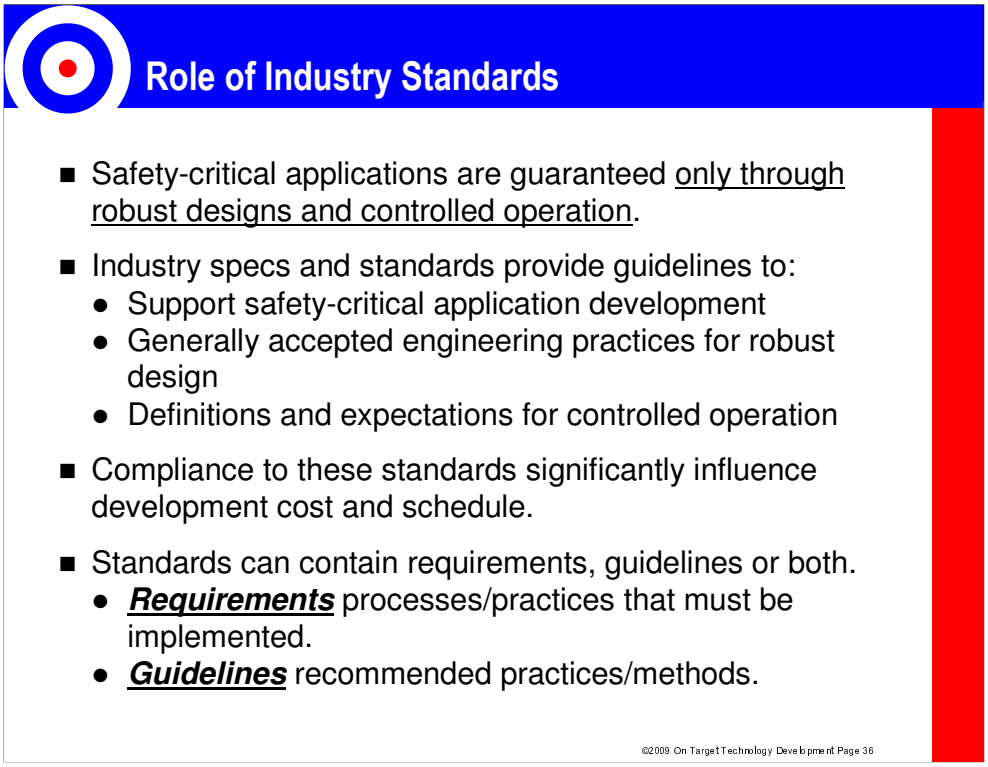


Fail-Active – This category is highly undesirable and strikes fear into the hearts of safety-critical systems designers. It occurs when mitigation into one of the other categories is not possible. It cannot be prevented completely, so applications typically allow a small probability, such as 10-9. In a fail-active state, actuators drive the system in an uncontrollable and unpredictable (nondeterministic) manner. Safety-critical systems are designed to minimize fail-active scenarios. *Example*: Automotive steer-by-wire applications have a unique challenge in that typical steering wheel interfaces have common mode failures that could inhibit steering control.



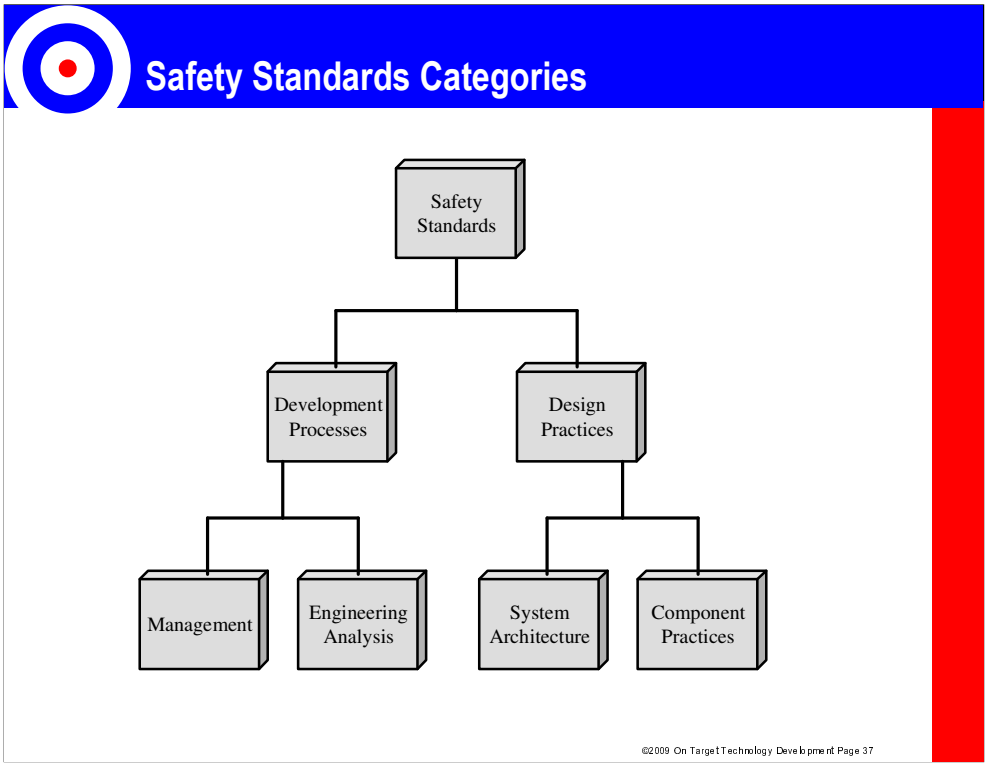
Application performance specifications may require system architectures to apply these degradation categories through multiple failure modes. For instance, after the first failure in a quad-redundant flight actuator system, the system may be required to remain fail-operational. After the second failure in the system, the application may require fail-passive operation. After three failures, the application may require fail-safe operation. System designers must be aware of failure response requirements and select architectures and components accordingly. In this application, the actuator system must be able to adequately move the aircraft surfaces with only two operating channels. Therefore, the actuators must be sized to enable operation with only two functioning actuators.


Industry Specifications and Standards
©2009 On Target Technology Development Page 35



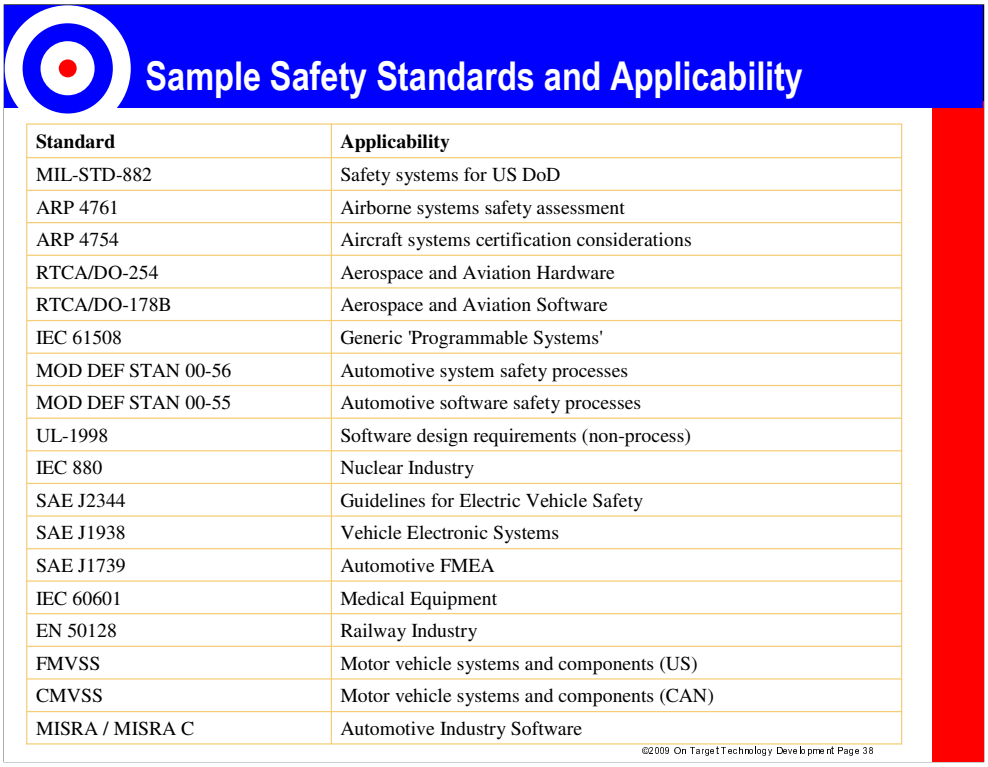
We claimed earlier that safety-critical applications are guaranteed only through robust designs and controlled operation. Industry specifications and standards provide generally accepted engineering practices and guidelines to support safety-critical application development. Development programs will identify applicable industry specifications and standards early in the life-cycle, as compliance to these standards can have a significant influence on development cost and schedule.

Standards can contain requirements, guidelines or both. Requirements are processes or design practices that must be implemented. Guidelines provide recommended practices and methods for satisfying requirements.

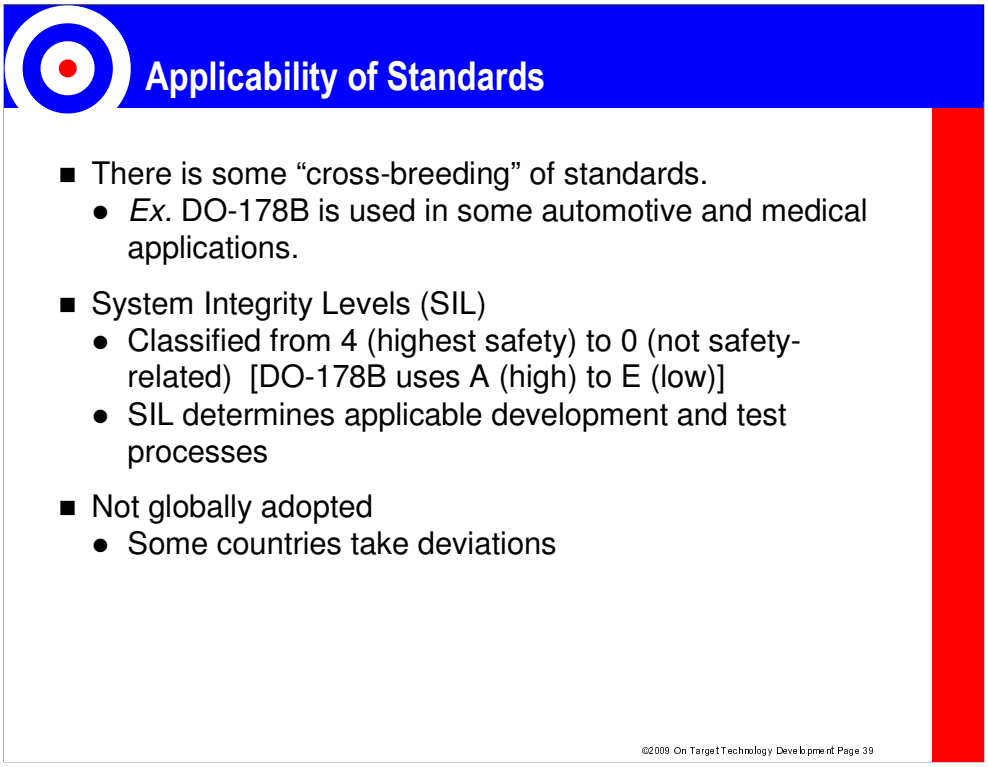


Standards can be applied to development processes and/or design implementation

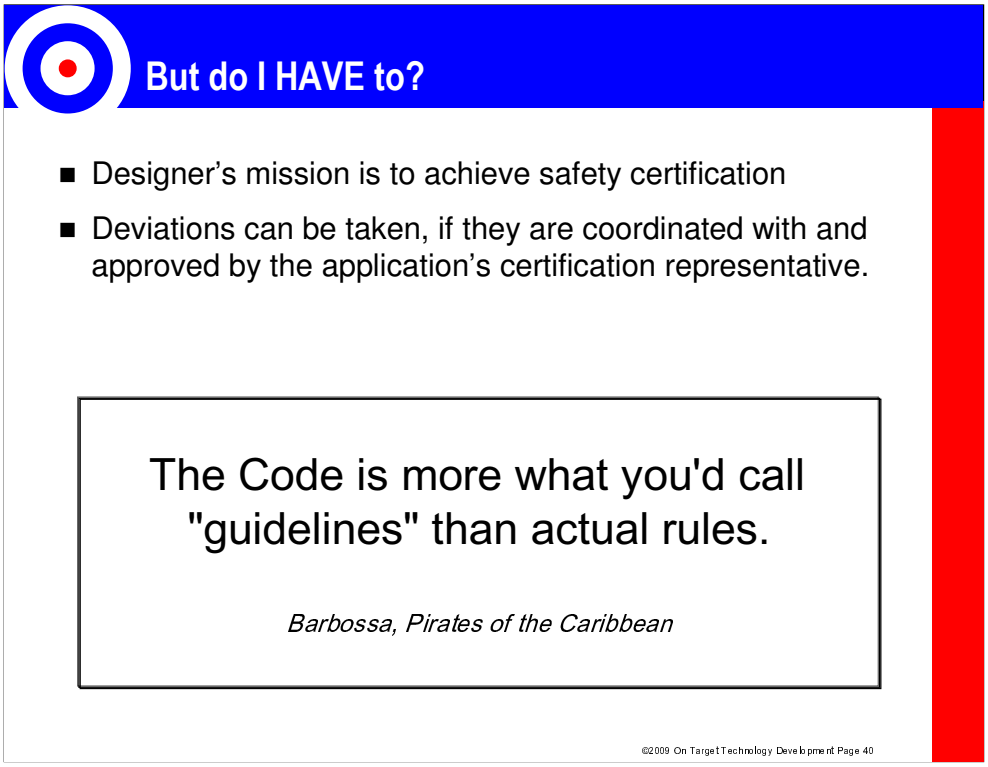
Examples of development processes are software development plans, quality management processes and verification plans. Engineering analysis standards include reliability analysis and other performance analyses. Design practices are very specific to the application industry. For instance, the Federal Motor Vehicle Safety Standards (FMVSS) provide design practices for motor vehicles.



Although the scope of this paper is not to provide a tutorial of safety standards, it is relevant to identify some standards and their applicable industries and applications. Table 1 highlights some common standards.

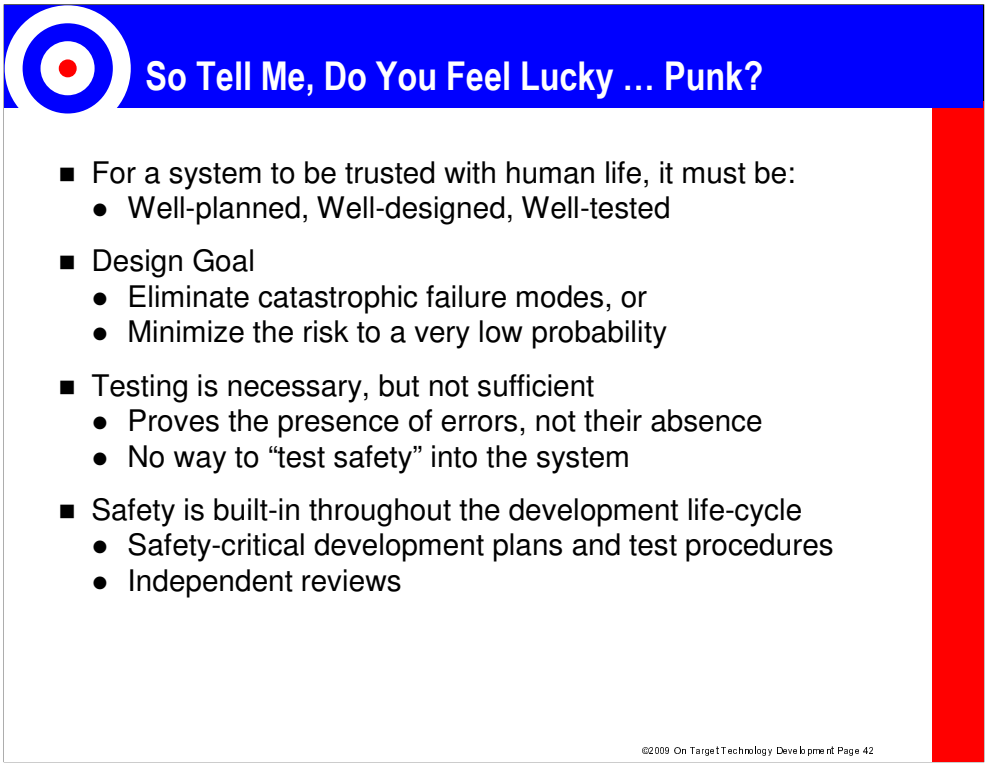


Many of the standards listed above classify systems into five System Integrity Levels (SILs) from 4 (highest safety) to 0 (not safety related). DO-178B uses A (highest safety) to E (lowest safety). The SIL of the development project drives the processes for that project. For instance, to test software in a DO-178B Level A project, such as a flight control system with no mechanical backup, all code statements must be executed, all code branches must be followed, and actual target hardware must be used.



To steal a quote from the movie *Pirates of the Caribbean*, many of these standards are "more what you'd call 'guidelines' than actual rules". The designer's real mission is to achieve safety certification. Deviations can be taken, if they are coordinated with and approved by the application's certification representative.


Development Strategies
©2009 On Target Technology Development Page 41

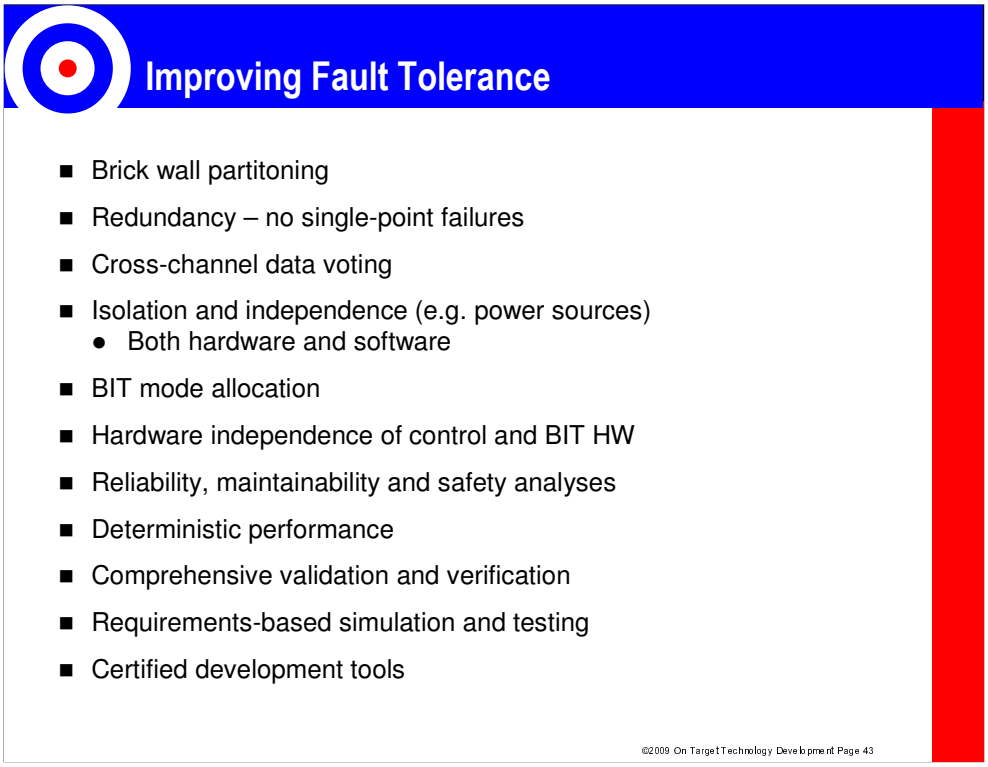


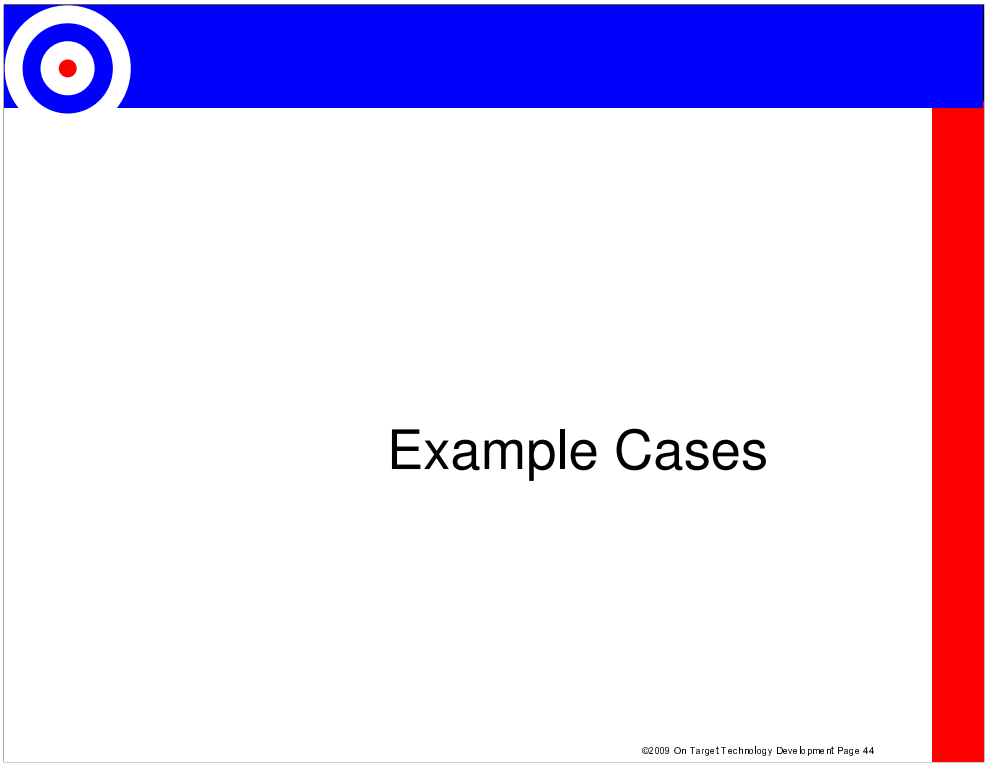
Systems that are well-planned, designed, tested and certified can be trusted with human life. The design goal of safety-critical system development is to eliminate failure modes that could result in catastrophic failures, or if that is not feasible, minimize the risk such that there is a very low probability of catastrophic failure.

Safety is proven not only through testing, but also through safe planning and development processes. Testing is a necessary, but not sufficient, element of safety-critical development – it only proves the presence of errors, not their absence. We haven't yet found a way to "test safety" into the system.

Safety must be built-in throughout the development life-cycle using safety-critical development plans and procedures, independent reviews, and comprehensive test cases.

This section highlights some basic development strategies that have proven to be important in many of the author's designs. It is by no means a comprehensive list, but it does capture some thought processes that must be exercised.


Improving Fault Tolerance
Brick wall partitoning
Redundancy – no single-point failures
Cross-channel data voting
Isolation and independence (e.g. power sources)
Both hardware and software
BIT mode allocation
Hardware independence of control and BIT HW
Reliability, maintainability and safety analyses
Deterministic performance
Comprehensive validation and verification
Requirements-based simulation and testing
Certified development tools
©2009 On Target Technology Development Page 43



Throughout my career, I have worked on safety-critical applications on a variety of platforms, including airplanes, locomotives, heavy-duty trucks, passenger vehicles, power systems, and medical diagnostics. Interestingly, the concepts needed to make these systems fault-tolerant, fail-operative or fail-safe are quite common.

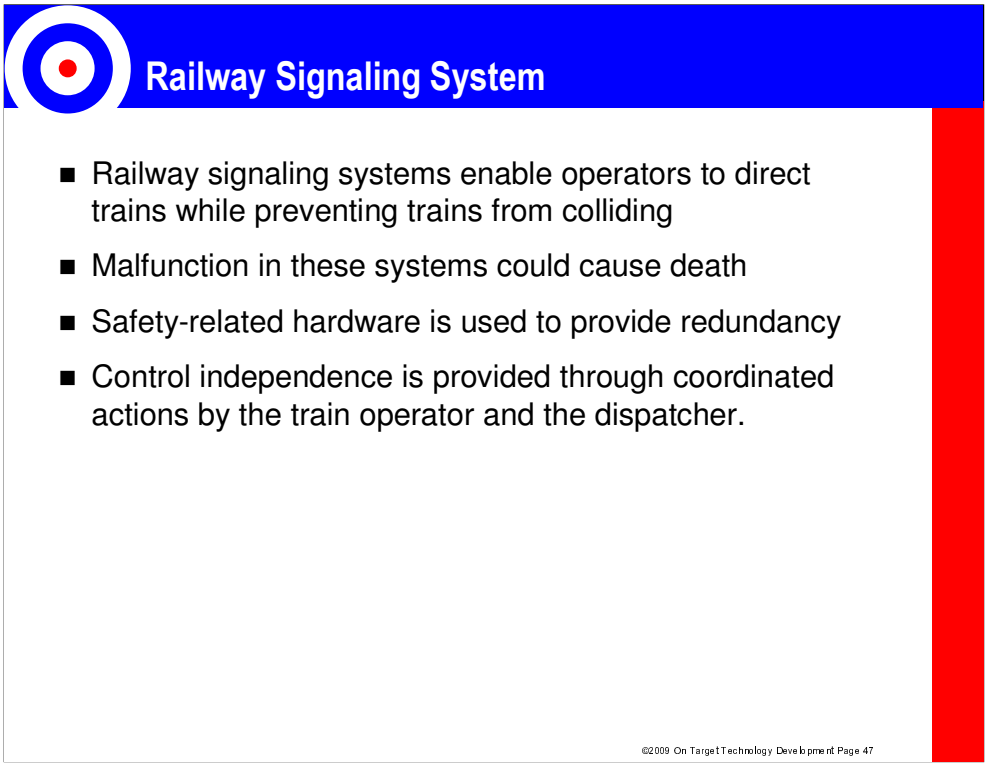
The following describes some examples of safety-critical systems.

Fly-by-wire Aircraft Control Systems

- In the past, many aircraft had mechanical backups for their electronic flight control systems.
- Fly-by-wire systems with no mechanical backups are becoming more prevalent in both military and commercial environments.
- Actuation systems, such as those that move flight surfaces, have become safety-critical.

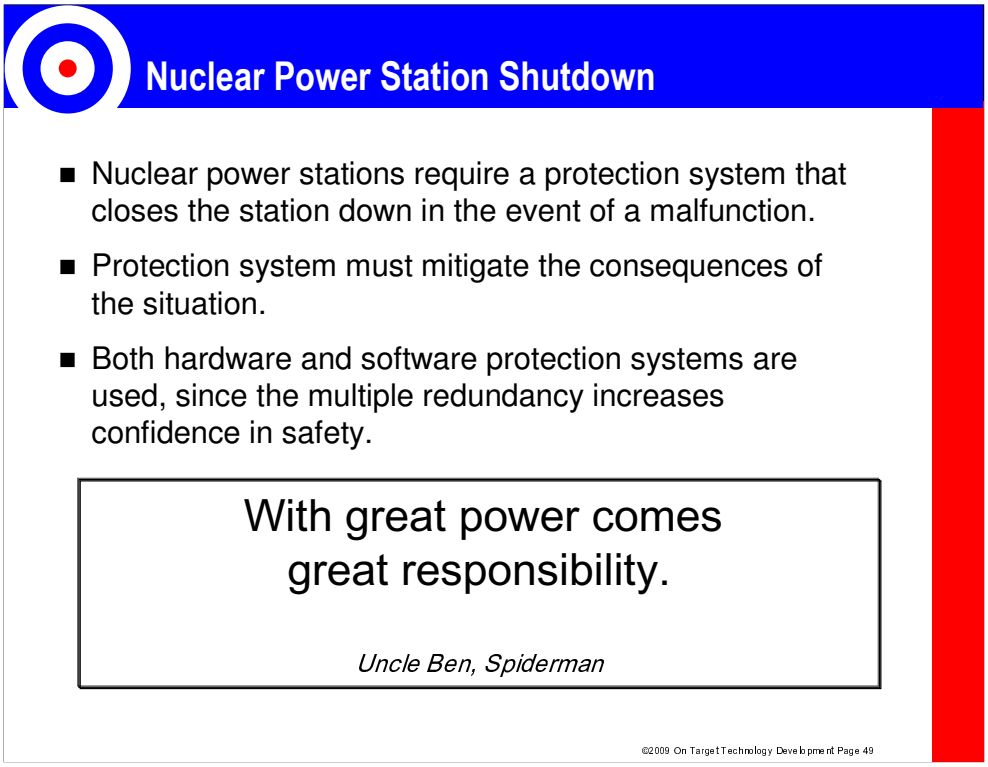
Automotive Drive-by-wire System

- Now developing full authority drive-by-wire systems:
  - Brake-by-wire
  - Steer-by-wire
  - Throttle-by-wire.
- Previously achieved fault tolerance through hardware redundancy.
- Redundancy can be cost and space prohibitive in drive-by-wire applications.
- Software standards such as MISRA are being developed to facilitate fault tolerance with less hardware redundancy.


Railway Signaling System
Railway signaling systems enable operators to direct trains while preventing trains from colliding
Malfunction in these systems could cause death
Safety-related hardware is used to provide redundancy
Control independence is provided through coordinated actions by the train operator and the dispatcher.
©2009 On Target Technology Development Page 47

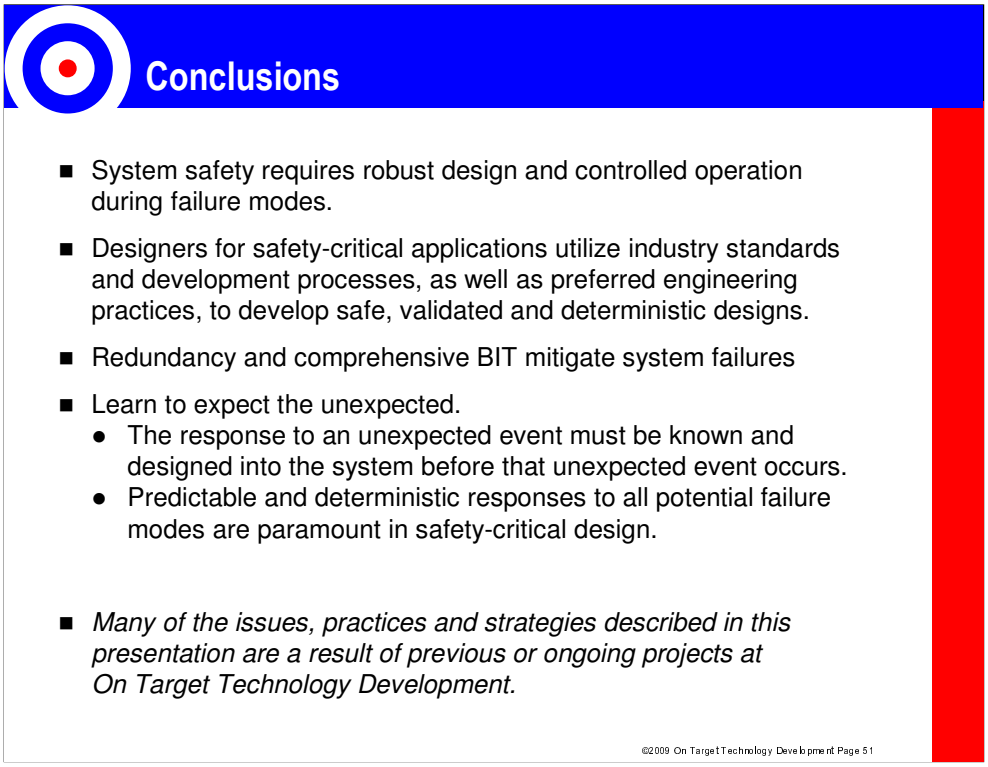
Submarine Depth Control System

- Similar challenges as aircraft applications
- Preferred fail-safe response depends on the application and environment.
- In a battle environment, underwater concealment is important, so surfacing the submarine may not be the best mitigation
- Submarine pilot must be able to eventually surface a failed vehicle to recover personnel


Nuclear Power Station Shutdown
Nuclear power stations require a protection system that closes the station down in the event of a malfunction.
Protection system must mitigate the consequences of the situation.
Both hardware and software protection systems are used, since the multiple redundancy increases confidence in safety.
With great power comes great responsibility.
Uncle Ben, Spiderman
©2009 On Target Technology Development Page 49

Medical Systems

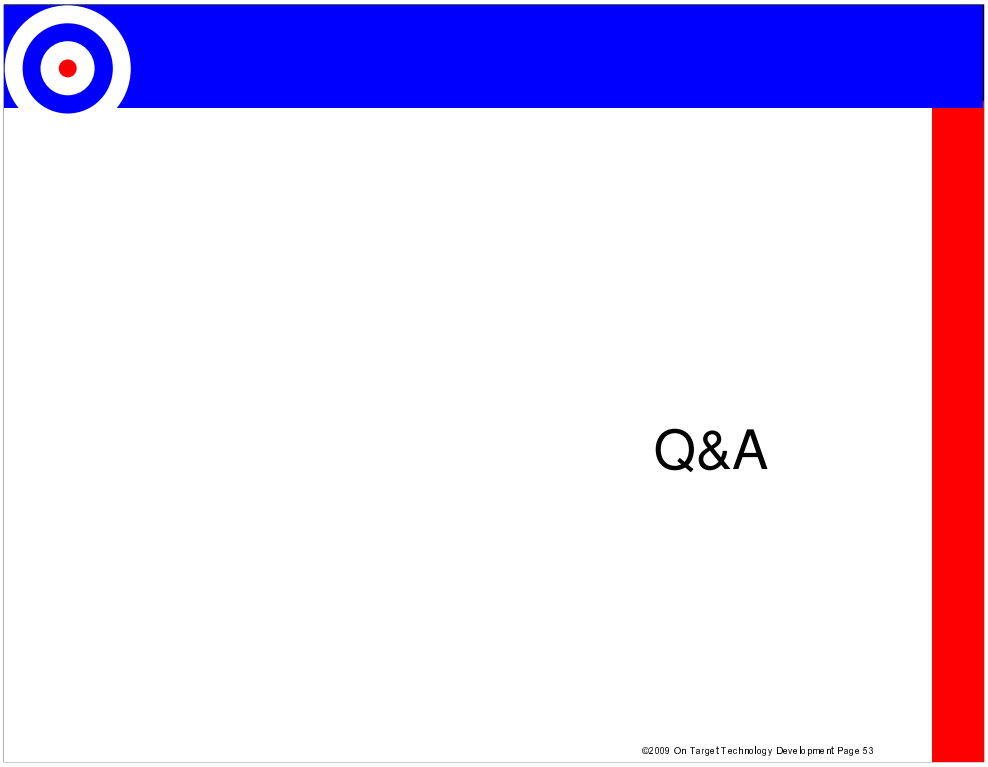
- Can be directly responsible for human life
- Examples:
  - Hardware that controls laparoscopic surgery devices
  - Software that monitors safe amounts of x-rays
  - Information systems that doctors use to coordinate medication
- Safety development standards have recently become a strong focus in the medical industry.
- Therac-25 radiation therapy machine has resulted in lost lives due to software-related radiation overdoses.



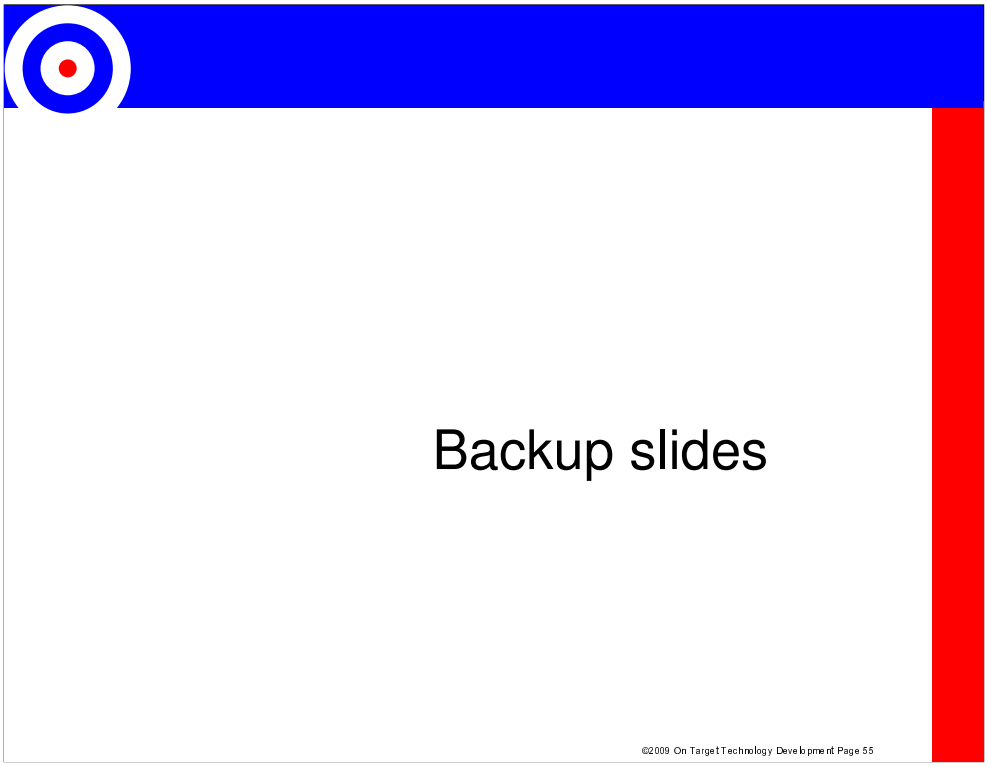
If we all keep our wits about us as we develop these safety-critical systems, we can sit back and relax in our confined space onboard an aircraft without worrying that a design failure will send our steel horse for a loop – even when we fly over the Equator.

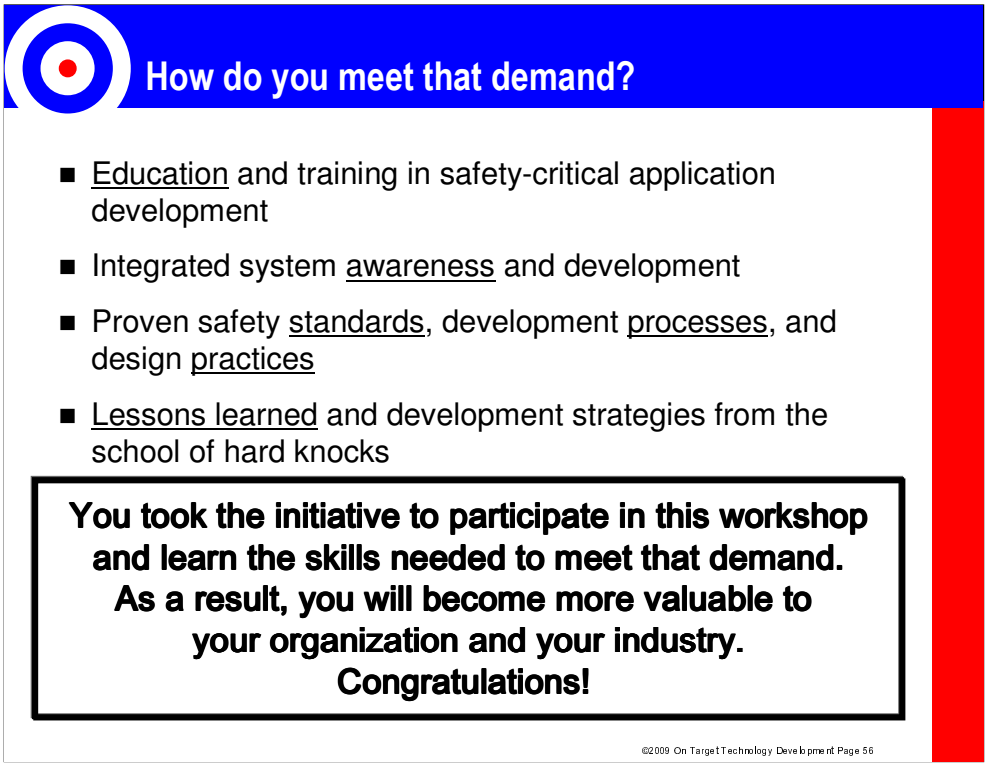
On Target Medical Actuator Projects

- Products, research projects, partnerships
- Medical monitor
- Muscle diagnostic system
- Circulation Actuation through the Limb Footwear (CALF)
- Personal transporter
- Respiratory therapy stimulation vest
- X-ray controller


Q&A
©2009 On Target Technology Development Page 53

If you have any further questions…


Backup slides
©2009 On Target Technology Development Page 55



There is only so much we can learn in a 90-minute course. But we will cover the highlights, and you will take that overview as a foundation for your future learning.

F-16 Thunderbird Crashes

Event #1 – Pilot Error

See http://www.f-16.net/f-16_news_article968.html

Event #2 – System Failure

F-16C performed a belly landing on the Luke AFB runway on June 17, 2004 at approximately 1416.

The pilot of the F-16 declared an in-flight emergency when he could not extend the landing gear. After numerous attempts to extend it, the aircraft ran out of fuel and the pilot was forced to put the jet down on its belly.

After sliding to a fiery stop, the canopy was seen to go up and the pilot sprinted away from the burning aircraft. The pilot is thought to be ok, but the aircraft is most likely totaled.

See http://www.f-16.net/f-16_news_article1100.html

A pilot's error caused a Thunderbirds F-16C to crash shortly after takeoff during a September airshow at Mountain Home Air Force Base, Idaho. The pilot ejected just before the aircraft impacted the ground.

On Wednesday the 21st, the Air Force Accident Investigation Board held a news conference at the home of the Thunderbirds - Nellis Air Force Base - to announce what caused an F16 to crash last September.

According to the accident investigation board report the pilot, 31-year-old Captain Chris Stricklin, misinterpreted the altitude required to complete the "Split S" maneuver. He made his calculation based on an incorrect mean-sea-level altitude of the airfield. The pilot incorrectly climbed to 1,670 feet above ground level instead of 2,500 feet before initiating the pull down to the Split S maneuver.

When he realized something was wrong, the pilot put maximum back stick pressure and rolled slightly left to ensure the aircraft would impact away from the crowd should he have to eject. He ejected when the aircraft was 140 feet above ground - just 0.8 seconds prior to impact. He sustained only minor injuries from the ejection. There was no other damage to military or civilian property.

The aircraft, valued at about $20.4 million, was destroyed.

The difference in altitudes at Nellis and Mountain Home may have contributed to the pilot's error. The airfield at Nellis is at 2,000 feet whereas the one at Mountain Home is at 3,000 feet. It appears that the pilot reverted back to his Nellis habit pattern for s aplit second. Thunderbird commander Lt. Col. Richard McSpadden said Stricklin had performed the stunt around 200 times, at different altitudes during his year as a Thunderbird pilot.

McSpadden says Stricklin is an exceptional officer. "He is an extremely talented pilot. He came in here and made an honest mistake," says Lt. Col. McSpadden. But that mistake has cost Stricklin his prestigious spot on the Thunderbird team. "He's assigned to Washington D.C.," says McSpadden. "He's working in the Pentagon there in one of the agencies."

The maneuver the pilot was trying to complete is called the "Split S Maneuver." The stunt requires that the pilot climb to 2,500 feet. Investigators say Stricklin only climbed to 1,670 feet before he went into the spinning roll.

The board determined other factors substantially contributed to creating the opportunity for the error including the requirement to convert sea level altitude information from the F-16 instruments - to their altitude above ground and call out that information to a safety operator below.

But the Air Force has now changed that as a result of the crash. Thunderbird pilots will now call out the MSL (mean-sea-level) altitudes as opposed to the AGL (above-ground-level) altitudes.

Thunderbird pilots will now also climb an extra 1000 feet before performing the Split S Maneuver to prevent another mistake like the one on Sep.14, 2003 from happening again.

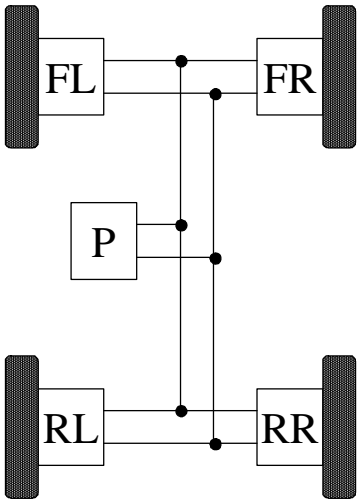
Captain Chris Stricklin has been in the Air Force since 1994 and flew with the Thunderbirds for the first season now. He has logged a total of 1,500+ flight hours and has received numerous awards. He served as a flight examiner, flight instructor and flight commander.

The Thunderbirds will again take to the skies this year. They have 65 air shows scheduled.

The September crash was the second involving a Thunderbirds jet since the team began using F-16s in 1983.

Pilot error was blamed for a Feb. 14, 1994, training crash involving in a maneuver called a spiral descent at the Indian Springs Auxiliary Airfield, northwest of Las Vegas. The pilot survived, but the maneuver was discontinued.

Automotive Brake-by-Wire System



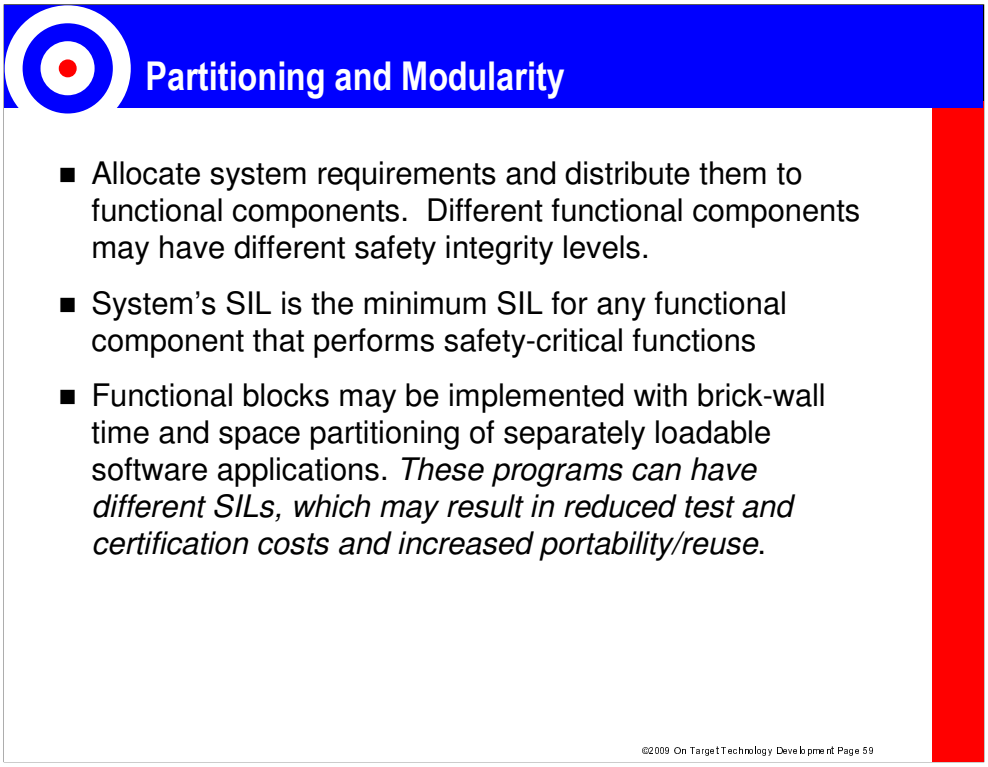
- A brake actuator interfaces with each wheel (FL, FR, RL, and RR).
- These actuators are controlled using the brake pedal input (P).
- Dual-redundant signal paths are used to prevent single-point failures in the wiring.

What happens when a brake fails?

(active/inactive)

A pedal?

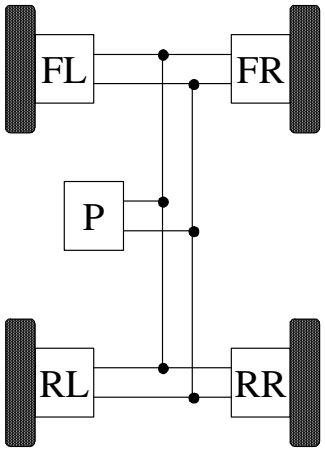
Brake-by-wire systems are safety critical systems. If the brakes fail to operate correctly, the vehicle will not be controllable. Imagine what would happen if you were driving 120 kph and the brakes would not actuate? What if the failure forced the brakes to actuate uncontrollably?

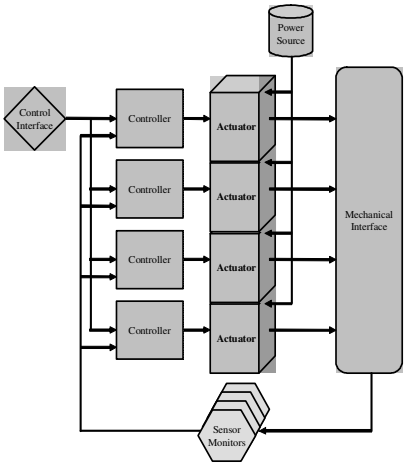


Remember - Selection of the safety integrity level is important because it determines which development guidelines, processes and standards will be required in development.

System Redundancy

- Primary means to develop fault tolerant systems of components
- Let's compare these two systems …



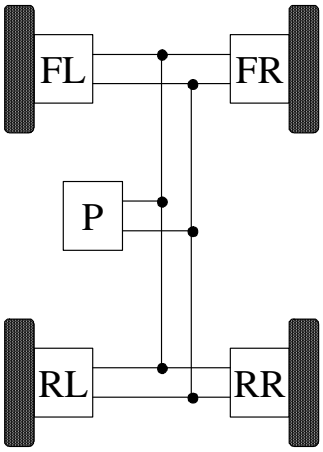


The first example is a brake-by-wire system. If one brake actuator fails in an inactive state (i.e. the brake cannot be applied), the brake actuators on the other three wheels will likely stop the vehicle. Redundancy allows the system to perform, albeit in a degraded mode, when failures occur. If a brake fails in an active state (i.e. the brake is applied when not commanded), the safety of the operator may be jeopardized, especially if it occurs while the vehicle is traveling at high speeds when the failure occurs. The Federal Motor Vehicle Safety Standards (FMVSS) provide guidelines to consider when developing this type of system. Depending on your application, the fail-safe position of brake actuators may be full-active, partially active, or full inactive.

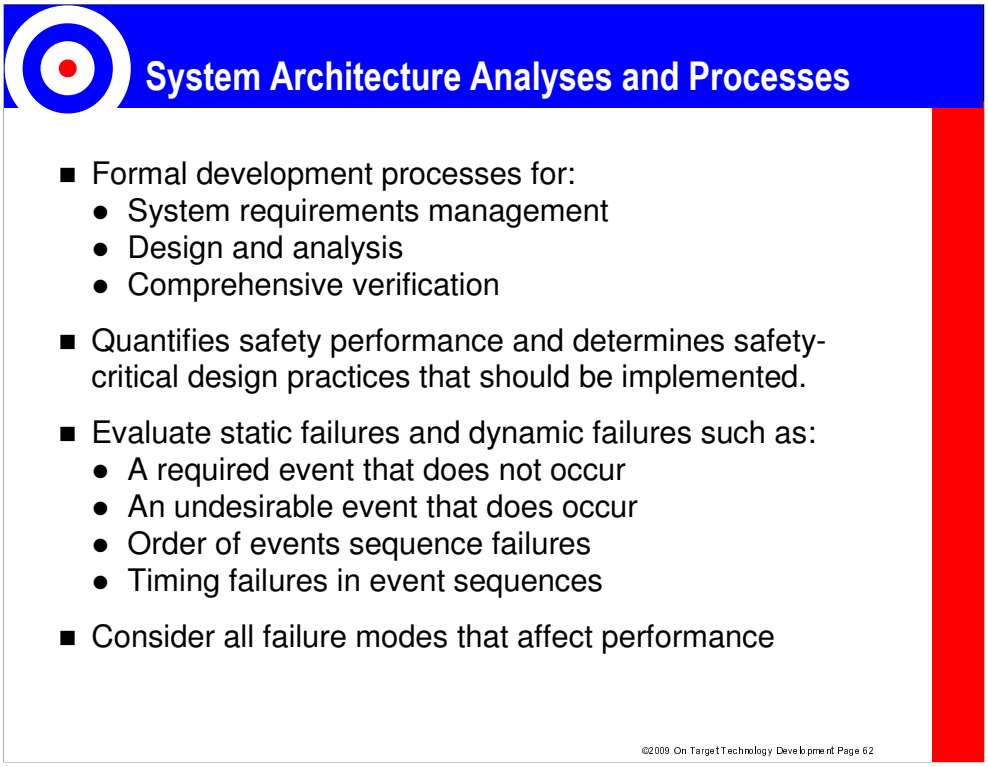
The second example shows a quad actuator system whereby all actuators control the same mechanical interface. Failures in the controllers, actuators and feedback sensors are mitigated by redundant components. However, failures in the control interface, power source and even the mechanical interface cannot be mitigated. These are called single-point failure modes, in which a failure at a single point will cause the system to be inoperable. Good system architectures minimize single-point failure modes, and ideally have none.

System Redundancy

- Brake-by-wire system



- If one brake actuator fails in an inactive state, the brake actuators on the other three wheels will likely stop the vehicle. Redundancy allows the system to perform (degraded) when failures occur.
- If a brake fails in an active state, the safety of the operator may be jeopardized.
- The Federal Motor Vehicle Safety Standards (FMVSS) provide guidelines to consider when developing this type of system.
- Depending on your application, the fail-safe position of brake actuators may be full-active, partially active, or full inactive.



The industry standards described earlier stress formal planning, development and test processes. Systems engineering for safety-critical systems emphasizes formal development processes for system requirements management, iterative design and analysis, and comprehensive verification. These guidelines, coupled with robust design practices, provide safe system architectures.

A safety analysis, such as the system safety assessment process described by guidelines like ARP4761 [5], are performed on the system architecture to quantify safety performance and ascertain what safety-critical design practices should be implemented.

Safety analyses for embedded systems evaluate not only static failures, but also dynamic failures. For example, embedded controllers can exhibit various system safety failures including but not limited to [6]:

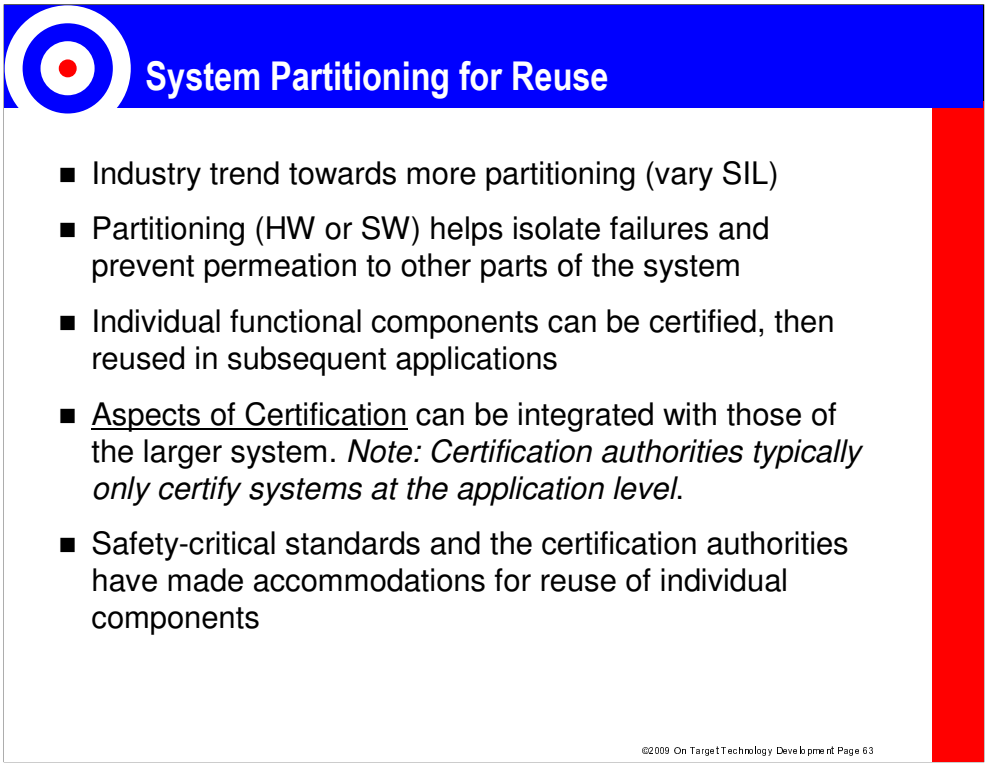
A required event that does not occur

An undesirable event that does occur

Order of events sequence failures

Timing failures in event sequences (maximum and minimum time constraints)

Without making claims of the relevance of Murphy's Law in engineering designs, a systems engineer for a safety-critical application must assume all possible failure modes will occur. Through robust design processes, inspection, analysis, design or test, the engineer must quantify that the probability of catastrophic failure meets application requirements (e.g. 10-9).

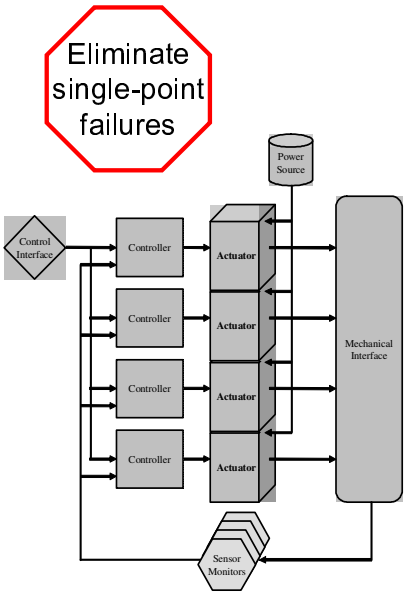


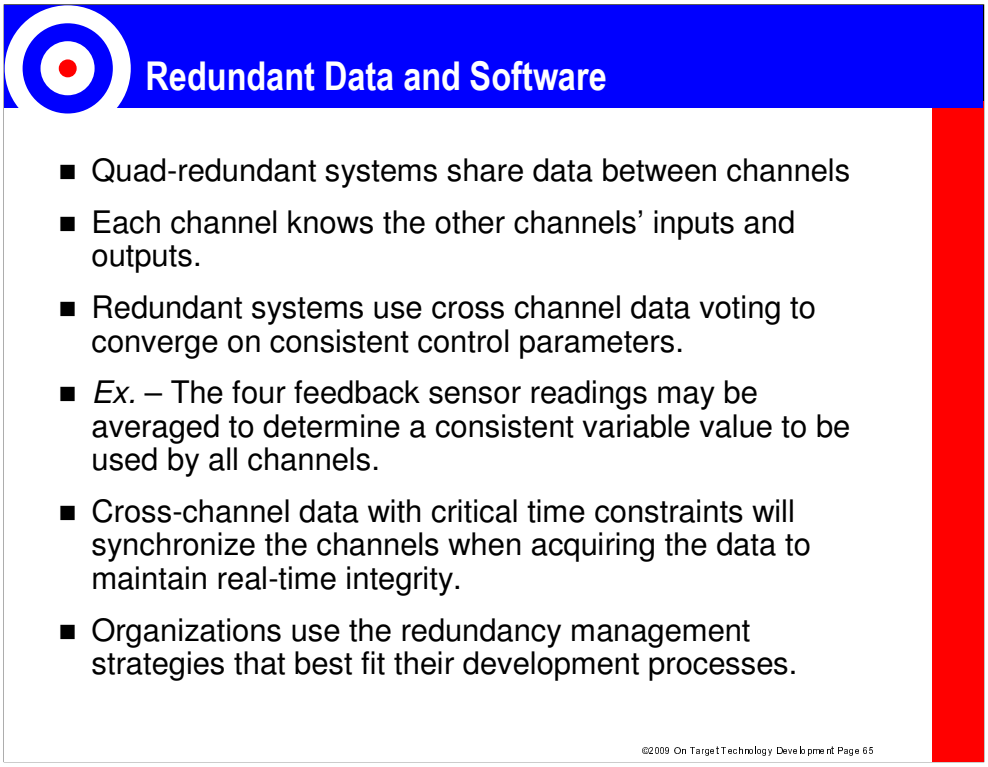
As systems become more complex, the trend is to make system architectures more modular. This enables individual functional components to be certified in applications, and then reused in subsequent applications. When those components are integrated into a larger system, their aspects of certification can be integrated with those of the larger system.

Certification authorities, such as the FAA, will typically only certify systems at the application level. For example, a flight control system is certified for an aircraft, not as a stand-alone application. However, safety-critical standards and the certification authorities have made accommodations for reuse of individual components. DO-178B includes section 12.1 to consider the use of previously developed software.

System Redundancy

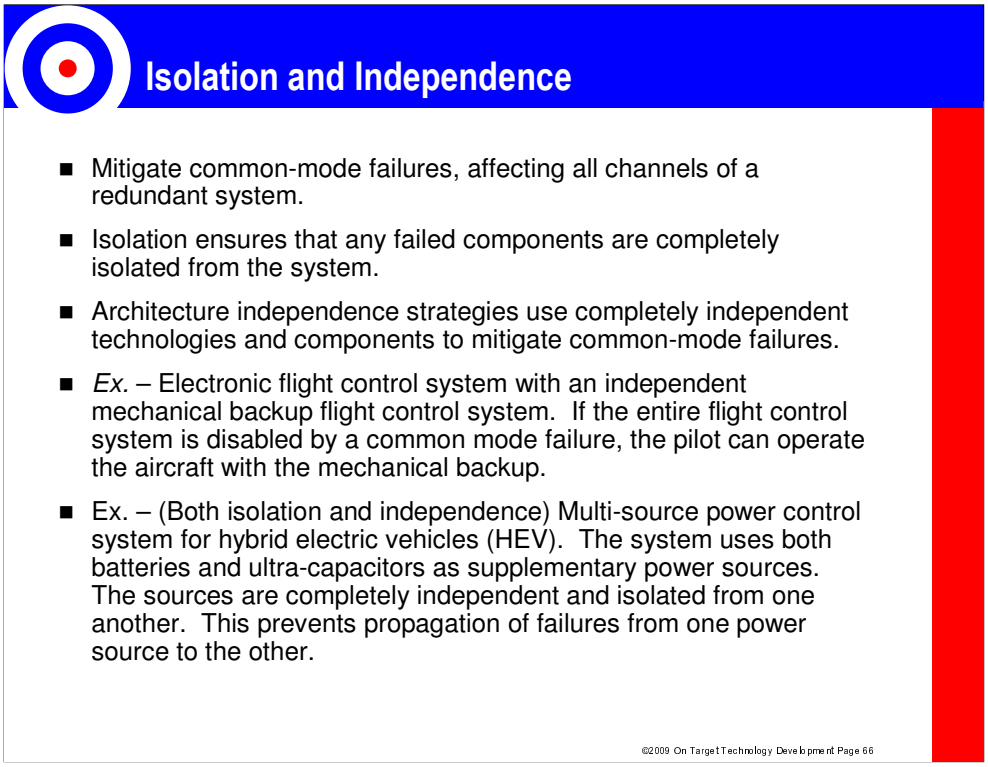
- Quad actuator system
- All actuators control the same mechanical interface
- Failures in the controllers, actuators and feedback sensors are mitigated by redundant components.
- Failures in the control interface, power source and even the mechanical interface cannot be mitigated.
- These are called single-point failure modes, in which a failure at a single point will cause the system to be inoperable.
- Good system architectures minimize single-point failure modes, and ideally have none.





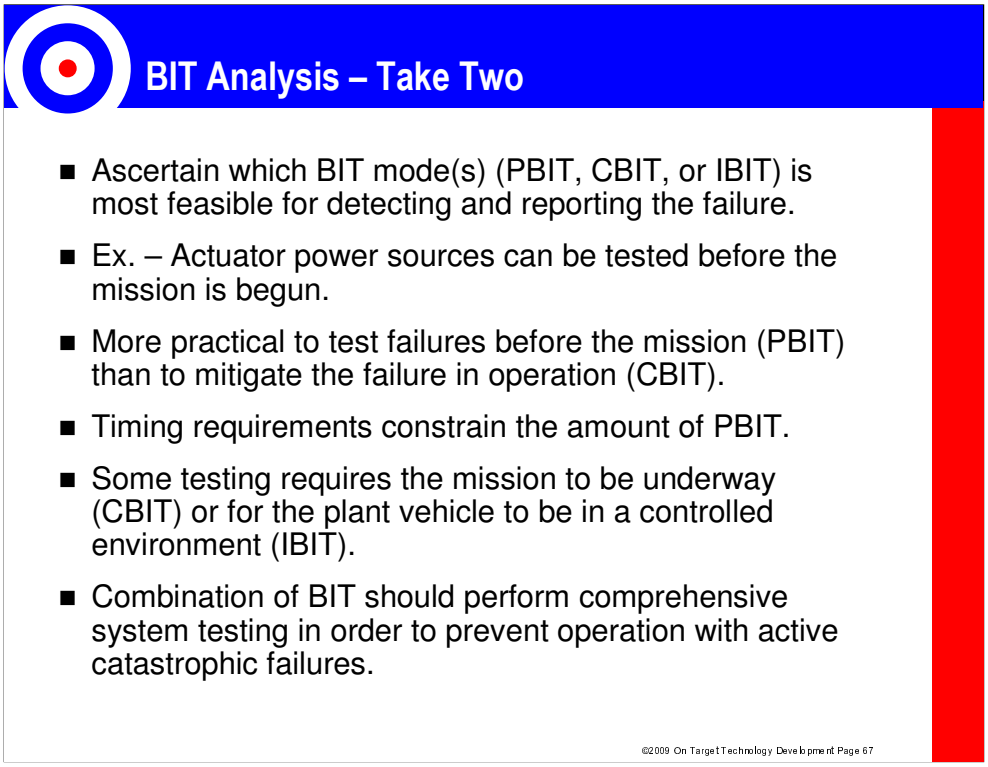
Redundancy practices are not only concerned with hardware redundancy, but also data and software redundancy. Quad-redundant systems usually share their data among the channels. Each channel understands what the other channels are reading as inputs and trying to control as outputs. Redundant systems use cross channel data voting to converge on consistent control parameters. For instance, the four feedback sensor readings may be averaged to determine a consistent variable value to be used by all channels. Cross-channel data that has critical time constraints will synchronize the channels when acquiring the data in order to maintain real-time integrity. The preferred strategies of redundancy management, as this practice is called, are as opinionated as your favorite pizza. Engineering organizations use the redundancy management strategies that best fit their development processes.

Redundancy is one means of improving the system architecture reliability. There are other best practices that are described in detail throughout the industry specifications and standards.

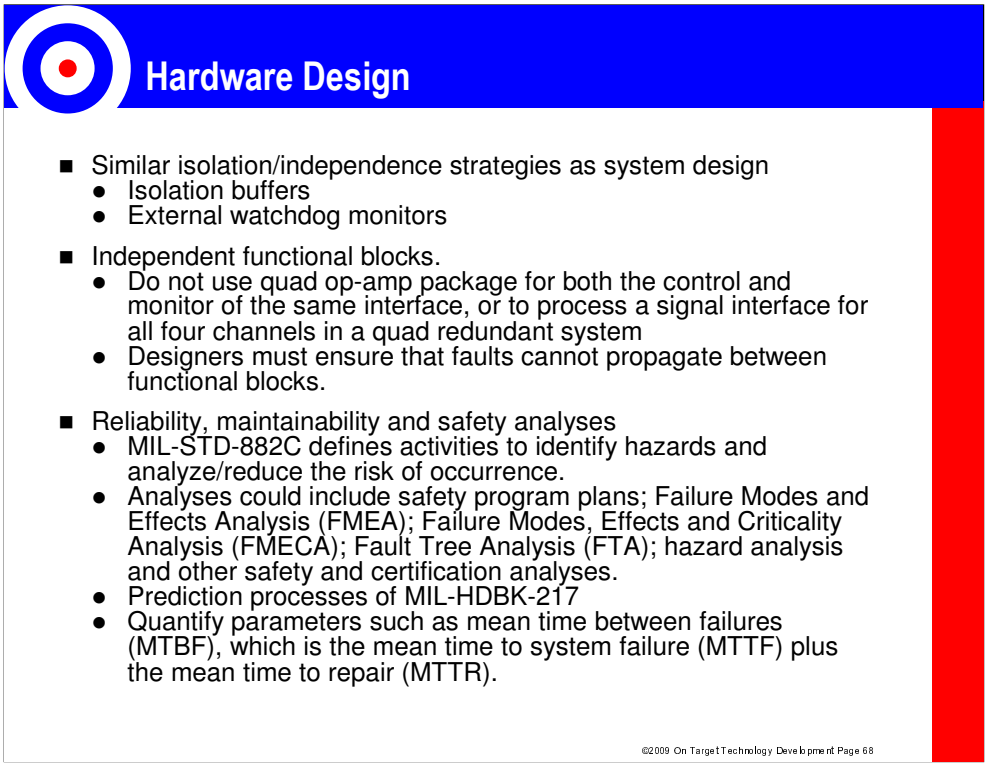


Whereas redundancy is concerned with adding redundant components to maintain performance; isolation and independence go even further by mitigating common mode failures, which can affect all channels of a redundant system. Isolation ensures that any failed components are completely isolated from the system. Architecture independence strategies use completely independent technologies and components to mitigate system failures. An example is an aircraft with an electronic flight control system and an independent mechanical backup flight control system. Even if the entire flight control system is disabled by a common mode failure, the pilot will be able to operate the aircraft with the mechanical backup system.

An example of an architecture that uses both isolation and independence is a multi-source power control system for hybrid electric vehicles (HEV). The system uses both batteries and ultra-capacitors as supplementary power sources. The sources are completely independent and isolated from one another. This prevents propagation of failures from one power source to the other.



Another important element of architecture development is the BIT analysis discussed earlier. When engineers evaluate the failure modes, they must ascertain which BIT mode(s) (PBIT, CBIT, or IBIT) is most feasible for detecting and reporting the failure. For instance, actuator power sources can be tested before the mission is begun. It is more practical to test failures before the mission begins (PBIT) than to try to mitigate the failure in operation. Powerup timing requirements may constrain the amount of testing performed during PBIT. Some failure testing may require the mission to be underway (CBIT) or for the plant vehicle to be in a controlled environment (IBIT). The combination of these BIT modes should perform comprehensive system testing in order to prevent operation with active catastrophic failures.



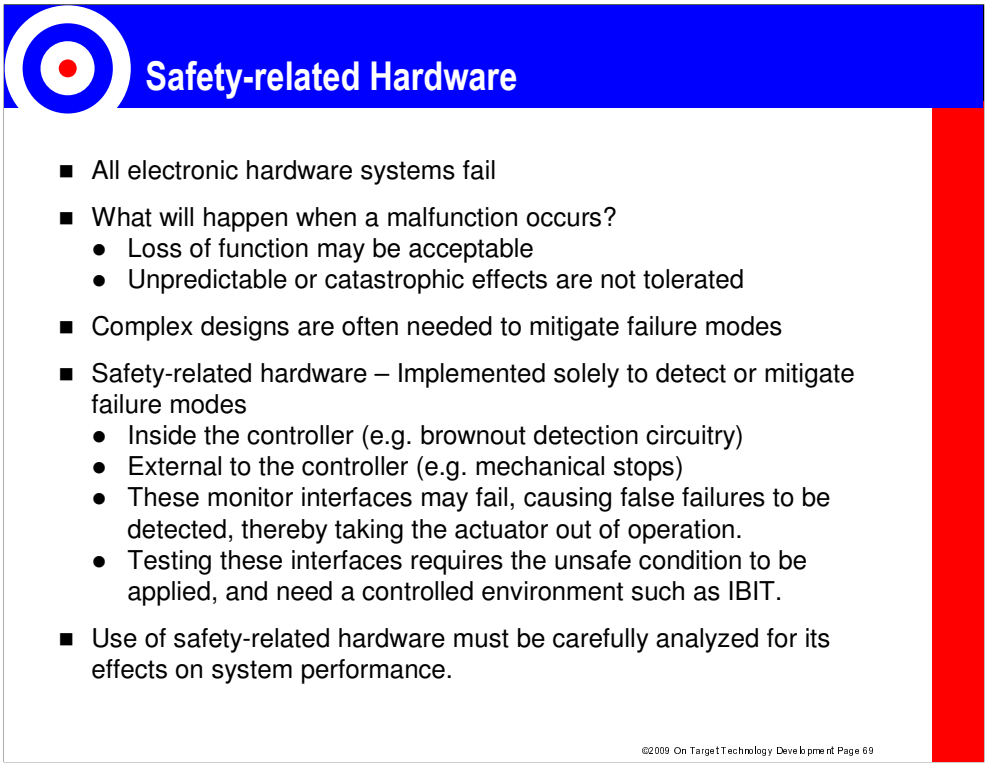
Many of the development strategies for system components also apply to hardware components. For example, hardware designers may choose to employ buffers to isolate signals or external watchdog monitors to provide independent processor integrity validation.

Hardware independence must also consider the allocation of components to functional blocks. For instance, a quad op-amp package should not be used for both the control and monitor of the same interface. In this case, a component failure will affect both the control and the monitor interfaces. Similarly, it should not be used to process a signal interface for all four channels in a quad redundant system. Designers must ensure that faults cannot propagate between functional blocks.

Reliability, maintainability and safety analyses are prevalent in safety-critical hardware design. MIL-STD-882C defines activities needed to identify possible hazards and analyze/reduce the risk of occurrence in use. These analyses could include safety program plans; Failure Modes and Effects Analysis (FMEA); Failure Modes, Effects and Criticality Analysis (FMECA); Fault Tree Analysis (FTA); hazard analysis and other safety and certification analyses. Using these analysis techniques and the prediction processes of MIL-HDBK-217, reliability engineers consider all safety-related failure modes and quantify parameters such as mean time between failures (MTBF), which is the mean time to system failure (MTTF) plus the mean time to repair (MTTR).

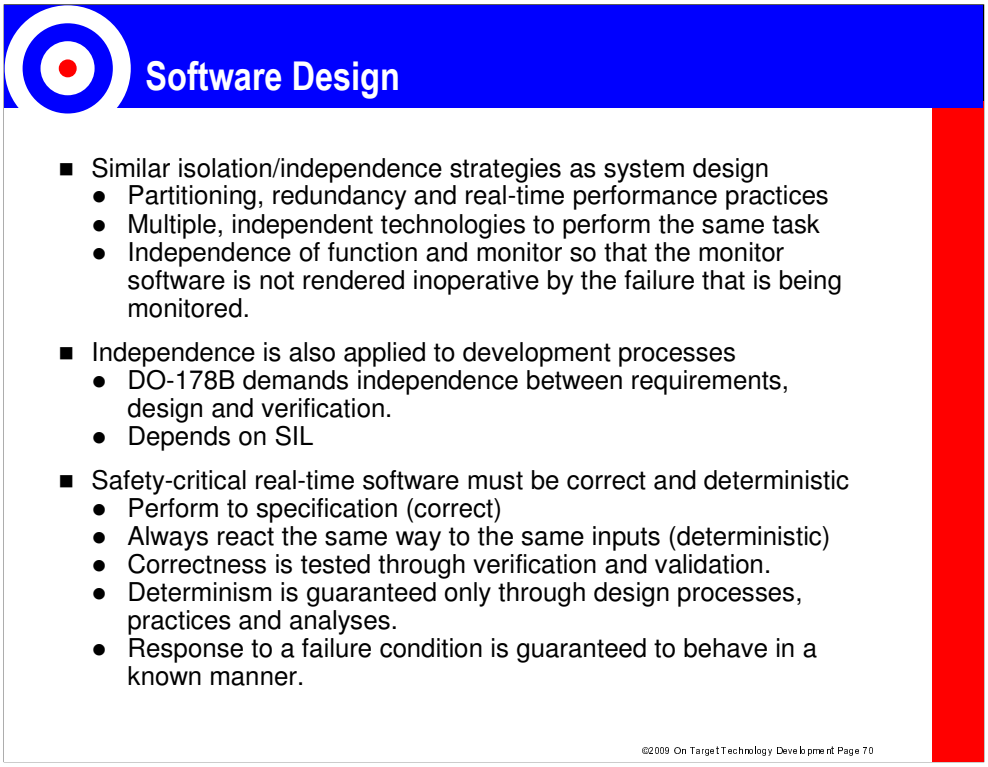
All electronic hardware systems fail at one time or another, so designers must understand what will happen when a malfunction occurs. A loss of function may be acceptable, but unpredictable or catastrophic effects are not tolerated.

Safety-related hardware, which is hardware that is implemented solely to detect or mitigate failure modes, requires careful design consideration. This type of hardware may reside inside the controller (e.g. brownout detection circuitry that shuts down the controller when power drops) or external to the controller (e.g. mechanical stops used to prevent an actuated swing arm from exceeding a safe operating range). A system risk is created when hardware is implemented solely for monitor purposes. These monitor interfaces may fail, causing false failures to be detected by the system, and thereby taking the actuator out of operation. Furthermore, testing that uses these interfaces typically requires the unsafe condition to be applied, and therefore can only be completed in a controlled environment such as IBIT. Use of safety-related hardware must be carefully analyzed for its effects on system performance.



All electronic hardware systems fail at one time or another, so designers must understand what will happen when a malfunction occurs. A loss of function may be acceptable, but unpredictable or catastrophic effects are not tolerated.

Safety-related hardware, which is hardware that is implemented solely to detect or mitigate failure modes, requires careful design consideration. This type of hardware may reside inside the controller (e.g. brownout detection circuitry that shuts down the controller when power drops) or external to the controller (e.g. mechanical stops used to prevent an actuated swing arm from exceeding a safe operating range). A system risk is created when hardware is implemented solely for monitor purposes. These monitor interfaces may fail, causing false failures to be detected by the system, and thereby taking the actuator out of operation. Furthermore, testing that uses these interfaces typically requires the unsafe condition to be applied, and therefore can only be completed in a controlled environment such as IBIT. Use of safety-related hardware must be carefully analyzed for its effects on system performance.



Many of the safety-critical system architecture development strategies are also applicable to safety-critical software designs. Partitioning, redundancy and real-time performance practices are employed in software architecture as well as system architectures. System independence and diversity can also be applied in software by using multiple, independent technologies to perform the same task. Software designs should also utilize independence of function and monitor so that the monitor software is not rendered inoperative by the failure that is being monitored.

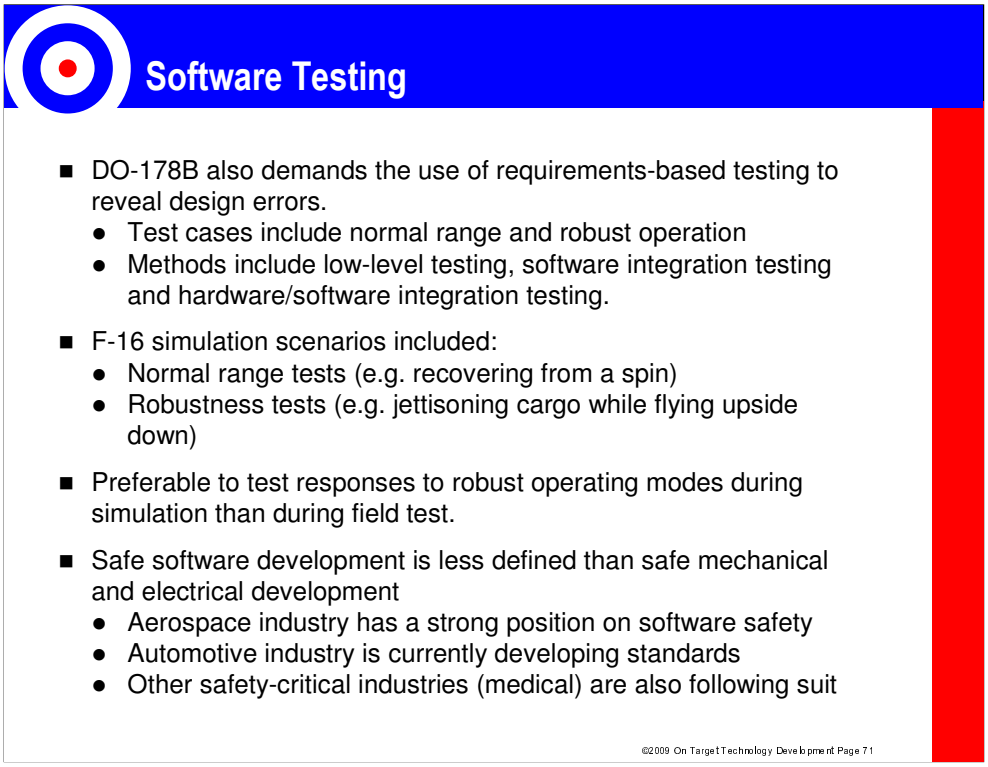
Independence is also applied to development processes. Depending on the safety integrity level, DO-178B [8] demands independence between requirements, design and verification. The engineer that designs a software function cannot be the same engineer that defined the requirements; and the engineer that verifies a software performance cannot be the same engineer that designed it.

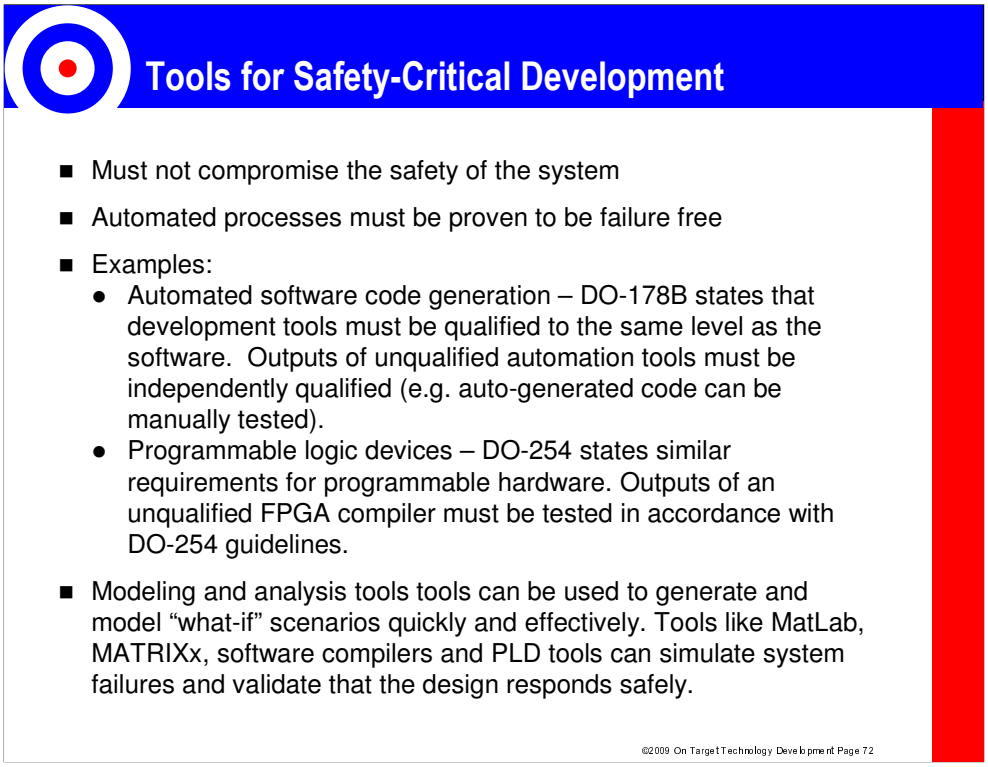
Safety-critical real-time software must be correct and deterministic. It must perform to specification (correct) and it must always react the same way to the same inputs (deterministic). Correctness is tested through verification and validation. Determinism is guaranteed only through design processes, practices and analyses. When software is deterministic, the response to a failure condition is guaranteed to behave in a known manner.

DO-178B also demands the use of requirements-based testing as an effective means to reveal design errors. Test cases should include normal range and robust operation. Methods may include low-level testing (i.e. unit testing with structural coverage), software integration testing and hardware/software integration testing.

Recall the F-16 simulation case discussed earlier. These scenarios include normal range tests (e.g. recovering from a spin) and robustness tests (e.g. jettisoning cargo while flying upside down). It is certainly preferable to test responses to these operating modes during simulation than during field test.

Much more is known about how to develop safe mechanical and electrical systems than safe software programs. To mitigate this issue, the aerospace industry has taken a strong position on software safety and the automotive industry is also building standards. Other safety-critical industries are also following suit, as can be seen in Table 1.


Software Testing
DO-178B also demands the use of requirements-based testing to reveal design errors.
Test cases include normal range and robust operation
Methods include low-level testing, software integration testing and hardware/software integration testing.
F-16 simulation scenarios included:
Normal range tests (e.g. recovering from a spin)
Robustness tests (e.g. jettisoning cargo while flying upside down)
Preferable to test responses to robust operating modes during simulation than during field test.
Safe software development is less defined than safe mechanical and electrical development
Aerospace industry has a strong position on software safety
Automotive industry is currently developing standards
Other safety-critical industries (medical) are also following suit
©2009 On Target Technology Development Page 71



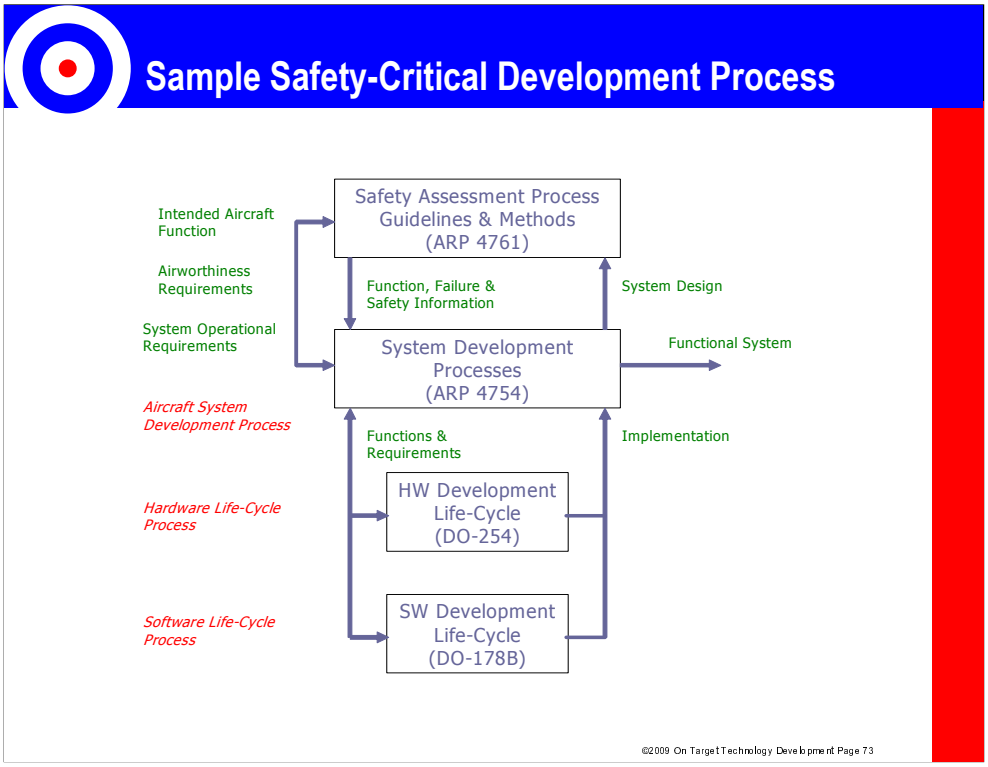
Tools used to assist in the development of safety-critical systems must not compromise the safety of the system. Any design or test automated processes must be proven to be failure free – either by the automated process itself or by subsequent testing or analysis.

Let's consider two examples:

Automated software code generation – DO-178B states that tools used for software development must be qualified to the same level as the software [8]. Unqualified automation tools may be used if appropriate qualification processes are exercised on the output. For instance, auto-generated code can be manually tested in accordance with DO-178B guidelines.

Programmable logic devices – DO-254 states similar requirements for programmable hardware [9]. Outputs of an unqualified FPGA compiler must be tested in accordance with DO-254 guidelines.

Lastly, do not underestimate the power of modeling and analysis tools to perform safety analyses and characterize the performance of safety-critical systems. These tools can be used to generate and model "what-if" scenarios quickly and effectively. As was seen in the F-16 simulation cases, it is much more effective to discover failures during simulation and test than in a fielded application. Tools like MatLab and MATRIXx can simulate system failures and validate that the design responds safely. Engineering analysis tools, such as those provided with software compilers and PLD design tools, can aid engineers in evaluating fault modes.



Design and development using these standards is an iterative process. There are interdependent relationships among the processes. Safety requirements are allocated by system development process to hardware and software. The hardware and software design performance analyses are used to verify requirements from the safety assessment process. The analyses continue until a system design is implemented that meets the safety and functional requirements of the application.