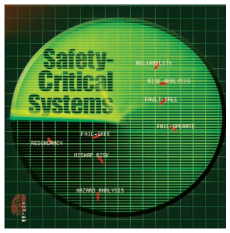
COVER FEATURE

Designing Safety-Critical Computer Systems



Designers must balance costs against an acceptable level of risk when implementing the techniques and devices that reduce the chance of mishap in safety-critical systems.

William R. Dunn
Independent Consultant

The ubiquitous computer is firmly established as the electronic component of choice for designing systems that control safety-critical applications. Such applications can be found everywhere: aircraft fly-by-wire controls, oil and chemical processing, hospital life-support systems, manufacturing robotics, and countless other commercial and industrial applications. As this century matures, developers will increasingly exploit computing's power in safety-critical applications that directly touch us all: steer-by-wire automotive systems, automated air- and surface-traffic control, powered prosthetics, and so on.

However, these computer-based systems raise the ongoing concern that they might fail and cause harm. Indeed, past computer failures have produced catastrophic results, most famously the notorious Therac 25, a therapeutic computer system intended to heal but which inadvertently killed and maimed patients before being forced off the market.[1]

The safety of computer-based systems is of long-standing and continuing interest to computing professionals. As research continues in this area, proposed system concepts and architectures—deemed safe by their developers—have been found to be impractical for real-life engineering applications that can place lives, property, or the environment at risk. Such dependable, seemingly safe, concepts and structures fail in practice for three primary reasons: Their originators or users

- have an incomplete understanding of what makes a system "safe,"
- fail to consider the larger system into which the implemented concept is to be embedded, or
- ignore single points of failure that will make the safe concept unsafe when put into practice.

Reviewing the fundamental definitions and concepts of system safety provides a framework for addressing these shortcomings. Exploring the systematic design of safety-critical computer systems in engineering practice helps to show how engineers can verify that these designs will be safe.[2]

DEFINING SAFE

The notion of safety is most likely to come to mind when we drive a car, fly on an airliner, or take an elevator ride. In each case, we are concerned with the threat of a *mishap*, which the US Department of Defense defines as an unplanned event or series of events that result in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment.[3]

The *mishap risk* assesses the impact of a mishap in terms of two primary concerns: its potential severity and the probability of its occurrence.[3] For example, an airliner crash would affect an individual more severely than an automobile fender-bender, but it's much less likely to happen. This assessment captures the important principle that systems such as cars, airliners, and nuclear plants are never absolutely safe. It also provides a design principle: Given our current knowledge, we can never eliminate the possibility of a mishap in a safety-critical system; we can only reduce the risk that it will occur.

Published by the IEEE Computer Society 0018-9162/03/$17.00 © 2003 IEEE

Risk reduction adds to system cost, however. Indeed, in some applications—such as nuclear energy—ensuring safety can dominate total system cost. When creating a safe system, minimizing this expense forces us to compromise to the extent that we expend resources to reduce mishap risk, but only to a level considered generally acceptable.

ACCEPTABLE MISHAP RISK

Generally, the public at large establishes the acceptable risk for a given mishap type in terms of its willingness to tolerate the mishap as long as it occurs infrequently. Statistics for various common mishaps and their average frequency represent acceptable risk—and can range anywhere from 10^{-2} to 10^{-10} incidents per hour. The relative rarity of such occurrences explains why, despite the tragic events underlying these statistics, most of us can feel relatively safe while driving a car or flying on an airliner.

This data can also give designers of safety-critical systems a sense of what constitutes safe and unsafe. For example, if they design a safety-critical computer system and project that it will have a 10 percent chance of catastrophic mishap per hour of operation, they know that the design is unsafe, and they must lower the mishap risk to an acceptable level.

Fortunately and wisely, system designers do not decide what constitutes an acceptable level. Instead, they rely on safety standards framed as public law or that result from the work of industrial associations, professional societies, and safety-related institutes that embody the general public's consensus of acceptable risk. For example, two widely used safety standards—the US government's Mil-Std-882D[3] and industry's IEC 61508[4]—provide detailed guidelines regarding acceptable risk.

THE COMPUTER SYSTEM

Typically, virtually any computer system—whether it's a fly-by-wire aircraft controller, an industrial robot, a radiation therapy machine, or an automotive antiskid system—contains five primary components.

The *application* is the physical entity that the system monitors and controls. Developers sometimes refer to an application as a plant or process. Typical applications include an aircraft in flight, a robotic arm, a human patient, and an automobile brake.

The *sensor* converts an application's measured physical property into a corresponding electrical signal for input into the computer. Developers sometimes refer to sensors as field instrumentation. Typical sensors include accelerometers, pressure transducers, and strain gauges.

The *effector* converts an electrical signal from the computer's output to a corresponding physical action that controls an application's function. Developers sometimes call an effector an actuator or final element. Typical effectors include motors, valves, brake mechanisms, and pumps.

The *operator* is the human or humans who monitor and activate the computer system in real time. Typical operators include an airplane pilot, plant operator, and medical technician.

System designers rely on safety standards that embody the general public's consensus of acceptable risk.

The *computer* consists of the hardware and software that use sensors and effectors to monitor and control the application in real time. The computer comes in many forms, such as a single board controller, programmable logic controller, airborne flight computer, or system on a chip. Many computer systems, such as those used for industrial supervisory control and data acquisition, consist of complex networks built from these basic components.

HAZARD ANALYSIS

The safety issues and design methodology associated with these networks and their complex structures strongly resemble those that apply to any simple computer system. Thus, we can study such a system to gain insights about basic design techniques that we would apply to more complex systems.

In the basic computer system, developers fully define the application, including all hardware, software, and operator functions that are not safety-related. Because the basic computer system employs no safety features, it probably will exhibit an unacceptably high level of mishap risk. When this occurs, solving the design problem requires modifying the operator, computer, sensor, and effector components to create a new system that will meet an acceptable level of mishap risk.

The design solution begins with the question, How can this basic computer system fail and precipitate a mishap? The key element connecting a failure in the basic system to a subsequent mishap is the *hazard*,[3] defined as any real or potential condition that can cause

- injury, illness, or death to personnel;
- damage to or loss of a system, equipment, or property; or
- damage to the environment.

Hazard examples include loss of flight control, nuclear core cooling, or the presence of toxic mate-

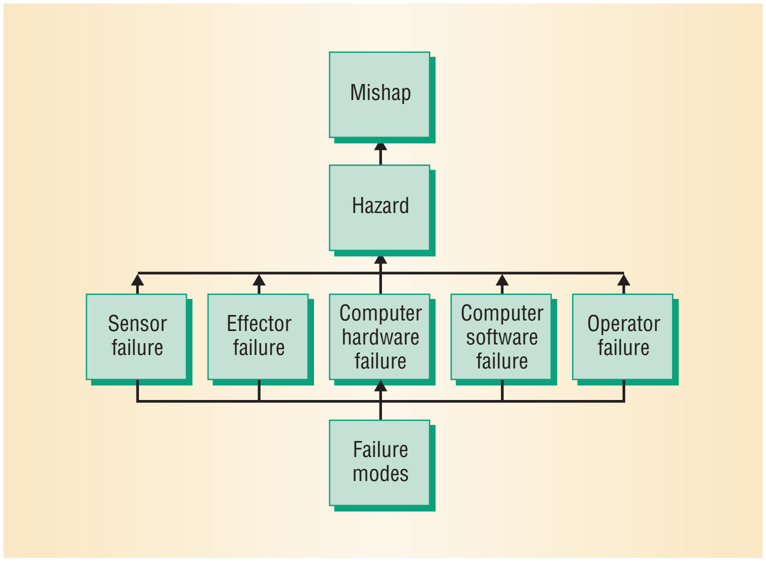


Figure 1. Mishap causes. System designers identify the application's attendant hazards to determine how system-component failures can result in mishaps.

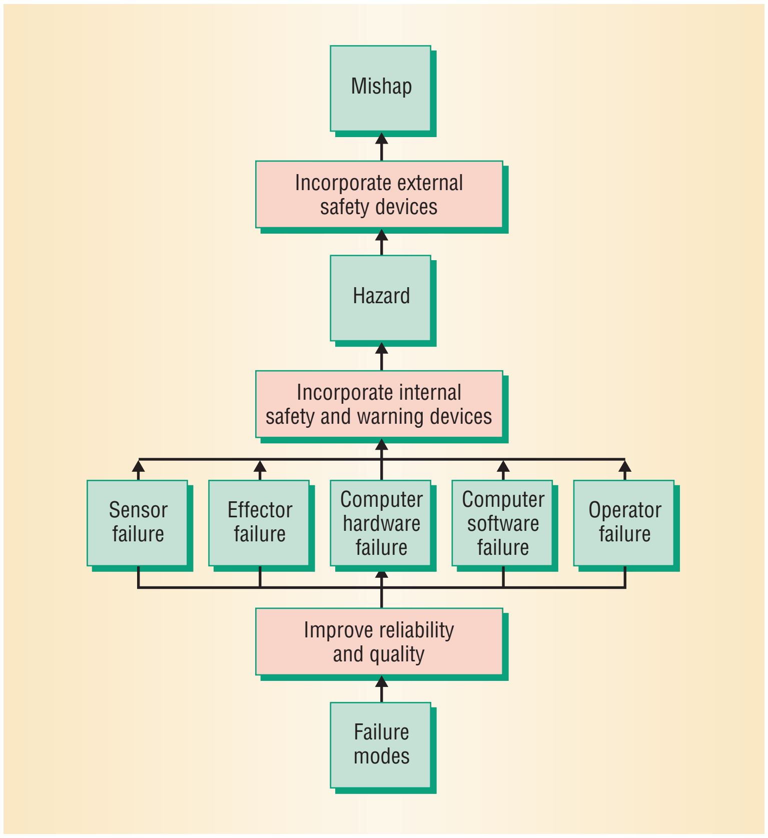

Mishap
Incorporate external safety devices
Hazard
Incorporate internal safety and warning devices
Sensor failure
Effector failure
Computer hardware failure
Computer software failure
Operator failure
Improve reliability and quality
Failure modes

Figure 2. Risk mitigation measures. Designers can modify a system to reduce its inherent risk by improving component reliability and quality and by incorporating internal or external safety and warning devices.

rial or natural gas. All such hazards reside in the application.

Thus, system design focuses first on the application component of the system to identify its attendant hazards. Then designers turn their attention to the operator, sensor, computer, and effector components. To determine how these components can fail and cause a mishap, the designers perform a failure-modes analysis to discover all possible failure sources in each component. These include random hardware failures, manufacturing defects, programming faults, environmental stresses, design errors, and maintenance mistakes.

These analyses provide information for use in establishing a connection between all possible component failure modes and mishaps, as Figure 1 shows. With this analytical background in place, actual design can begin.

A SIMPLE EXAMPLE

Consider a basic computer system used for electrically heating water. In this case, the application is a steel tank that contains water. The effector, a computer-controlled electric-heater unit, heats the water. A temperature sensor measures the water temperature and transmits a corresponding signal back to the computer. Software in the computer maintains the water temperature at 120°F by turning the heater on if the sensed temperature dips below this setting and by turning the heater off if the temperature climbs above the setting.

That the water this system stores might overheat presents one hazard. A potential mishap could occur if the water overheats to the boiling point and causes the tank to explode. Another potential mishap could occur if a person opens a water tap and the overheated water, under high pressure in the tank, scalds that individual as it exits the faucet and flashes into steam.

Several failures can create this hazard. The temperature sensor might fail and inaccurately signal a low temperature. The heater unit might fail and remain on permanently. Computer interface hardware might fail, permanently signaling an "on" state to the heater. A computer software fault, possibly originating in an unrelated routine, might change the set point to 320°F. The operator might program an incorrect set point. Component failures might also occur because of

- a maintenance error such as the repair person installing the wrong temperature sensor,
- an environmental condition such as the heater being placed in an overly warm environment that causes a chip failure, or
- a design failure that results in using the wrong sensor for the selected operating temperature.

This hot water system, as it stands, has an unacceptable risk of mishap.

MISHAP RISK MITIGATION

Given the system's high risk of mishap, design attention turns to modifying it to mitigate this risk. Designers can do this in three ways:

- improve component reliability and quality,
- incorporate internal safety and warning devices, and
- incorporate external safety devices.

Figure 2 shows how and where applying these mishap-risk-mitigation measures can alleviate the computer system mishap causes shown in Figure 1.

Improving reliability and quality involves two measures: improving component reliability and exercising quality measures that will avoid or eliminate the sources of component failure. Reliability improvement seeks to reduce the probability of component failure, which in turn will reduce mishap probability.

A widely used and effective approach for improving reliability employs redundant hardware and software components. Redesign can remove component reliability problems that stem from environmental conditions.

Other sources of component failure such as personnel error, design inadequacies, and procedural deficiencies are more elusive. IEC 61508 includes these sources of failure in a general category described as *systematic failures* and recommends various quality-oriented approaches for avoiding or eliminating them.

Although reliability and quality measures can reduce mishap risk, they normally will not lower it to an acceptable level because component failures will still occur. When a project requires additional risk mitigation steps, *internal safety devices* form the next line of defense. An example of an internal safety device is the thermocouple circuit, which shuts off the gas supply in a home heating furnace should its flame go out. Developers implement these devices in both hardware and software. Internal safety devices not only reduce the effects of hardware and software faults but also provide a barrier against systematic failures, including personnel errors, design inadequacies, and procedural deficiencies.

Even after designers have taken these measures, system failures can still occur, resulting in mishaps. *External safety devices*, which can range from simple physical containment through computer-based safety-instrumented systems, provide a last line of defense against these residual failures. These devices provide protection when the application experiences a hazardous event.

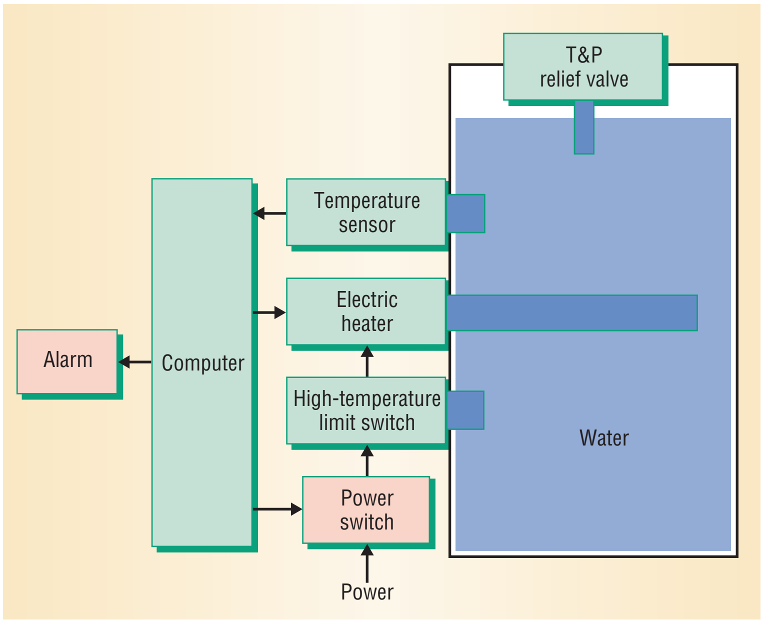


Figure 3. Applying risk-mitigation measures. The addition of safety devices such as a high-temperature limit switch and a temperature-and-pressure (T&P) relief valve has reduced the computer-controlled water heating system's operational risk.

To achieve effective mishap risk mitigation, developers usually strive to apply all three of these mitigation measures concurrently to create a layered approach to system protection. Because even the most lavish project has limited development resources, designers should apply all three types of risk mitigation in a balanced way to reduce mishap risk. In addition, risk mitigation efforts must be distributed evenly across the system's sensor, effector, computer, and operator components because a single neglected failure in any one part of the system can make the aggregate mishap risk totally unacceptable.

THE EXAMPLE REVISITED

Returning to the hot-water system example, upgrading the basic computer system to incorporate safety devices can reduce the system's risk. To reduce risk, the water-heater application uses all three of Figure 2's risk-mitigation measures in three protective layers.

Domestic water heater manufacturers generally employ hardware components with reliability superior to that of everyday household components. Manufacturers take extraordinary quality measures to assure the heater tank's structural integrity. Although these reliability and quality measures can reduce component failure probability and therefore mishap risk, they do not by themselves make the system safe—which is why heater manufacturers add both internal and external safety devices.

Figure 3 shows an internal safety device, the high-temperature limit switch. This device interrupts electric power to the heater when the water temperature, measured by an independent tem-

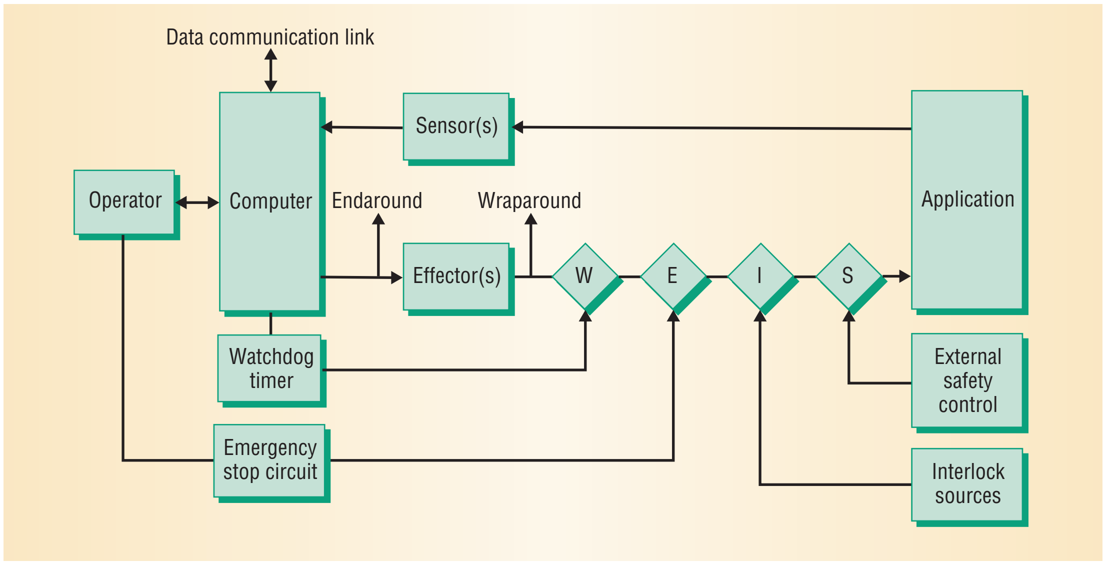


Figure 4. Risk mitigation methods. Designers have added several risk-mitigation devices to this system, including a watchdog timer, emergency stop circuit, and interlocks that inhibit effector actions unless specific external conditions are satisfied.

perature sensor, reaches a preset level. This temperature-limit switch thus provides protection for failures in any of the sensor, effector, computer, or operator components.

Additional internal safety protection can also be written into computer software. To give this software authority over hardware, the manufacturer interposes a power switch—which can be controlled by a computer output port—between the effector power source and the effector. The safety software can detect failures in the various components and cut power to the effector when it detects a failure.

These devices can thus further reduce mishap risk, but they still fall short of lowering it to an acceptable level. Manufacturers achieve additional risk protection by fitting an external safety device to the tank. The *temperature and pressure relief valve*—the T&P valve shown in Figure 3—is an external safety device that relieves tank overtemperature, which can lead to a scalding mishap, and overpressure, which can cause an explosion mishap.

ADDITIONAL SAFETY DEVICES

Figure 4 generalizes the water heater example by showing a basic computer system that has been modified to include risk-mitigation techniques found in real-life applications. One such technique, the emergency stop circuit, inhibits effector outputs by forcing the system into a safe state—as shown by the line in Figure 4 that connects the operator component to the diamond-enclosed E.

Systems often employ interlocks that will inhibit effector action unless some specific external physical conditions are satisfied. The switch that stops the cooking when a user opens a microwave oven door is one example of an interlock. As the water-heater example suggests, designers can reduce mishap risk in a system by using a computer to detect component failures and modifying effector controls to bring the system to a safe state. The design can incorporate various approaches to detecting failures in individual sensors, including reasonableness tests, informational redundancy, state estimators, and analytical redundancy.[2]

As Figure 4 shows, to detect effector failures, the design can use a wraparound in which the effector output feeds back into the computer to verify that the output matches the system command. The same basic approach uses endarounds to verify computer I/O integrity. When the system detects wraparound or endaround mismatches, it signals the effector to shift to a safe state. A failure in the forward computer-to-effector path may, however, prevent the shift. For this reason, developers usually build an additional, independent safety control into the system to neutralize the effector output when it detects wraparound or endaround mismatches.

Finally, most industrial controllers employ a watchdog timer circuit between the computer and effector output. The computer continuously refreshes this circuit with hardware- and software-generated electrical pulses. As long as these pulses continue, the circuit keeps the effector output connected to the application. If the pulses cease through hardware or software failure, the circuit times out, and the system inhibits further effector output.

FAIL-OPERATE SYSTEMS

In fail-safe systems, hardware, software, or an operator detects a failure and modifies effector output so that the system enters a safe, generally nonoperating state. Most real-world applications are fail-safe systems. Many computer systems, however, such as fly-by-wire aircraft control systems, must continue safe operation after one or more components have failed. These *fail-operate* computer systems achieve their fault-tolerance capability through redundancy.

One fail-operate approach uses a backup system that can take over the computer's safety-critical functions should the system fail. For example, computer-controlled fly-by-wire airliners, such as the Airbus A320 family, can fly—but at great risk—using primitive mechanical controls as a backup should the computer system fail.[5] As is the case in the Airbus application, a system's performance usually degrades when used in backup mode. A second approach simply replicates components so that if a given component fails, the system includes one or more duplicates to continue the required function.

Although component redundancy is a simple concept, the details of implementing it are not. First, the design must replicate virtually every critical component in a system, including computers, sensors, effectors, operators, power sources, and interconnects. Second, the design must incorporate a redundancy-management process into the fail-operate system's hardware, software, or operator components to detect failures when they occur, isolate the failed component, and reconfigure the system so that one or more healthy components will replace or mask the failed counterpart.

These failure, detection, isolation, and reconfiguration processes can quickly become complex, resulting in system development costs that far exceed those of the corresponding basic computer system.[2] For this reason, component redundancy becomes a practical design option only when a backup system is infeasible or when performance must be maintained following one or more component failures.

To design a fail-operate system, many developers use a two-step process in which they first select a redundant hardware structure or architecture and subsequently flesh out this framework with the appropriate redundancy management hardware and software processes. This two-step process is impractical, however, because the system's redundancy-management scheme—*not* its redundant structure—primarily governs the achievable risk level associated with a redundant computer system.

Consequently, designers must resort to a cut-and-try process that will meet a required risk level and, at the same time, satisfy the usual engineering economies of cost, power, weight, and so on. The preferred approach therefore begins with the basic, nonredundant system hardware structure and incrementally introduces redundancy and redundancy-management processes until a fail-operate system emerges that meets the desired safety goal.[2]

Fail-operate computer systems achieve their fault-tolerance capability through redundancy.

EVALUATING SAFETY-CRITICAL COMPUTER SYSTEMS

After the designers have applied measures to mitigate mishap risk to a basic system, they must determine if the modified system design meets an acceptable level of mishap risk. They can use three analytical techniques to make this determination.

In *failure modes and effects analysis* (FMEA), the designer or analyst looks at each component in the system, considers how that component can fail, then determines the effects each failure would have on the system.[2,6] This analysis seeks first to verify that there is no mishap-producing single point of failure in the system because such a potential point of failure would nullify the benefits of applying mitigation measures elsewhere in the system.

Fault tree analysis (FTA) reverses this process by starting with an identified mishap and working downward to identify all the components that can cause a mishap and all the safety devices that can mitigate it.[7,8] This downward decomposition process builds a graphical structure called a fault tree.

In contrast to FMEA and FTA, which are both qualitative methods, *risk analysis* (RA) is a quantitative measure that yields numerical probabilities of mishap.[2,7] To perform RA, the analyst must determine the component failure probabilities for the hardware, software, and operator components in the fault tree.[2,6,7] In accordance with standards such as Mil-Std-882D[3] and IEC 61508,[4] designers usually estimate failure probabilities on a per-hour basis.

If the system consists of redundant components, designers calculate its unreliability—the probability that it will not operate over the span of one hour. Next, they determine mitigation failure probabilities for the fault tree's hardware, software, and operator safety devices. If a mitigation device includes redundant components, designers determine its unavailability—the probability that it will not mitigate if required.

The designers assign these component- and mitigation-failure probabilities to elements in the fault tree, then propagate them upward to yield a figure for mishap risk. If this results in an unacceptable figure, they must implement additional mitigation measures. As a side benefit, the fault tree shows where to add these measures in the system. If, on the other hand, the risk calculation yields an acceptable result, the design is ready for additional validation steps[5] such as in-depth risk assessment, testing, and field trials to assure that the system, when implemented, will be safe.

Although it may seem obvious, a developer's concerns about a safety-critical system's continuing safety do not end with design and implementation. Indeed, a vigorous system safety program must be in place throughout the system's operational life to ensure that mishap risk is maintained at or below the level achieved in the original design.[3,4]

Achieving risk reduction in a new or existing computer system design requires dealing with *all* the system's components: hardware and software, sensors, effectors, operator, and—most importantly—the primary source of harmful energy or toxicity, the application. After identifying the application's hazards, determining component failure modes, and introducing appropriate risk mitigation measures, designers analyze the modified system to obtain a risk estimate. If risk remains unacceptable, they must take additional risk mitigation steps until the modified system has an acceptable level of mishap risk. ■

References

1. N. Leveson and C.S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, July 1993, pp. 18-41.
2. W.R. Dunn, *Practical Design of Safety-Critical Computer Systems*, Reliability Press, 2002.
3. *Standard Practice for System Safety*, Mil-Std-882D, US Dept. of Defense, 2000; http://www.geia.org/sstc/G48/882d.pdf.
4. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, IEC 61508, Int'l Electrotechnical Commission, 2000.
5. N. Storey, *Safety-Critical Computer Systems*, Addison-Wesley, 1996.
6. W. Goble, *Control Systems Safety Evaluation and Reliability*, ISA, 1998.
7. T. Bedford and R. Cooke, *Probabilistic Risk Analysis: Foundations and Methods*, Cambridge Univ. Press, 2001.
8. *Fault Tree Handbook*, NUREG-0492, US Nuclear Regulatory Commission, 1981; www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/ sr0492.pdf.

William R. Dunn, *formerly the director of the University of Southern Colorado's research station at NASA Ames Research Center, is now an independent consultant. His research and engineering interests include design and validation of safety-critical flight- and ground-based digital systems. Dunn received a PhD in electrical engineering from Santa Clara University. Contact him at wrdunn@syv.com.*