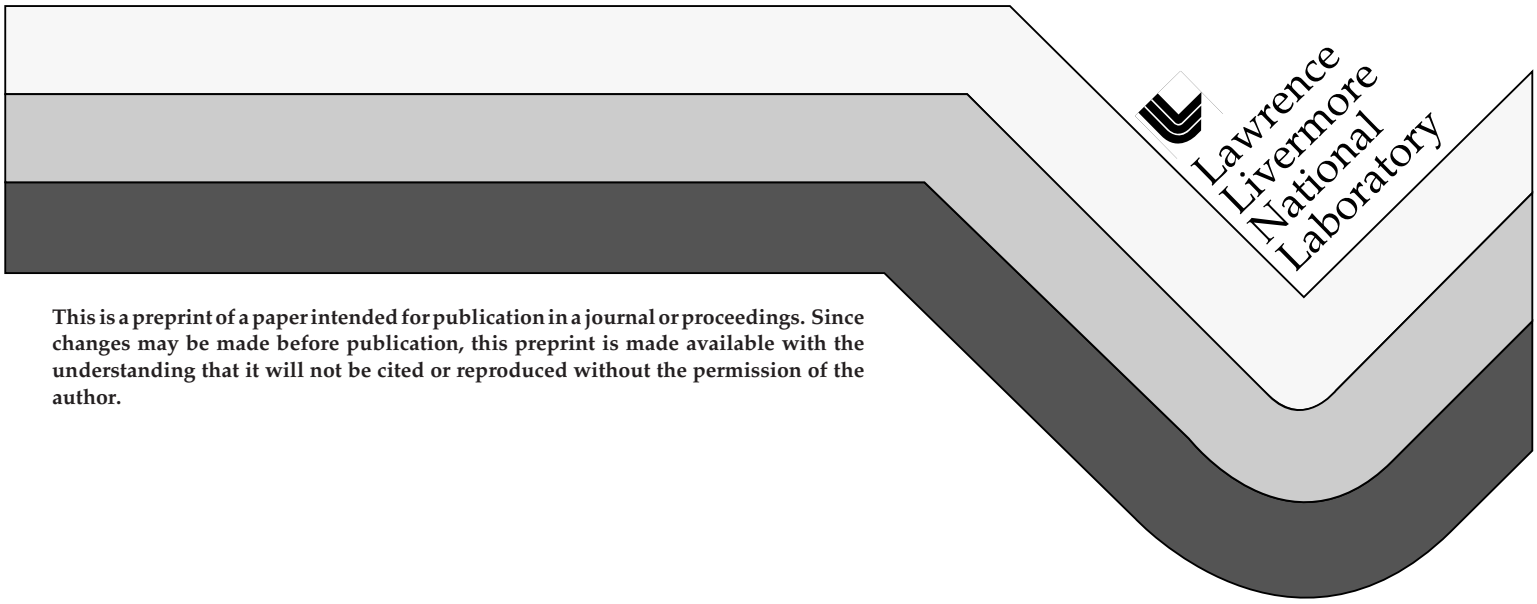
UCRL-JC-128414
PREPRINT

Defense Against Common-Mode Failures in Protection System Design

R. H. Wyman
G. L. Johnson

This paper was prepared for submittal to the
International Atomic Energy Agency Technical Committee Meeting on
"Advanced Technologies for Improving Availability and Reliability of Current and Future Water Cooled Nuclear Power Plants"
Argonne National Laboratory
Argonne, Illinois
September 8-11, 1997

August 27, 1997



This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Defense Against Common-Mode Failures in Protection System Design[1]

Robert H. Wyman and Gary L. Johnson, Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550

Abstract

The introduction of digital instrumentation and control into reactor safety systems creates a heightened concern about common-mode failure. This paper discusses the concern and methods to cope with the concern.

Common-mode failures have been a "fact-of-life" in existing systems. The informal introduction of defense-in-depth and diversity (D-in-D&D)—coupled with the fact that hardware common-mode failures are often distributed in time—has allowed systems to deal with past common-mode failures. However, identical software operating in identical redundant systems presents the potential for simultaneous failure. Consequently, the use of digital systems raises the concern about common-mode failure to a new level. A more methodical approach to mitigating common-mode failure is needed to address these concerns.

Purposeful introduction of D-in-D&D has been used as a defense against common-mode failure in reactor protection systems. At least two diverse systems are provided to mitigate any potential initiating event. Additionally, diverse displays and controls are provided to allow the operator to monitor plant status and manually initiate engineered safety features.

A special form of common-mode failure analysis called "defense-in-depth and diversity analysis" has been developed to identify possible common-mode failure vulnerabilities in digital systems. An overview of this analysis technique is provided.

Introduction

The term "common-mode failure" (CMF) has been in the reactor vocabulary for a number of years. It has been somewhat controversial because there were few concrete examples of CMFs; those few were not always accepted as having failed from some common mechanism at work. The problem stemmed from the way the failures occurred. There were no cases where a large number of "identical" devices failed at nearly the same time in nearly the same way. The failures occurred randomly over time and thus appeared to be ordinary random failures. It was only upon closer inspection that the common-mode failure became clear.

However, with the advent of software operated devices—where multiple redundant units would all be executing the same program with essentially the same inputs and outputs and more-or-less synchronous—the possibility of simultaneous failure in redundant units becomes all too real.

The purpose of this paper is to describe some of the issues of software-induced common-mode failures including assessment issues and methods of avoiding catastrophic failure in the event that such a common-mode problem exists.

[1] This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Common-Mode Failures

For the purposes of this paper, CMFs are failures attributable to a common cause. The results of a CMF may be the complete loss of a protective function at a critical time.

CMF mechanisms may be the result of requirement errors or omissions, design flaws, manufacturing problems, installation faults, application errors, quality control (QC) failures, etc. However, really good QC may actually enforce a CMF that is caused by other problems, in that good QC will force all items to be very similar if not identical.

Many CMFs give the appearance of random failures because the failures are random in time, although the mechanism of the failure is the same in all units. Variable environment and usage will spread the failures out in time so that they appear to be failures from random causes.

Examples

Experience in the industry has shown that, while not common, common-mode failures do in fact occur in hardware. Some examples follow.

Salem

The original reactor trip breakers were not specifically designed for the application in that they did not contain an undervoltage trip coil as such. A modification was made so that the trip mechanism was spring-operated and the spring was held by a relay or solenoid that was energized during normal operation. When the voltage fell below some value, the coil would release the spring tripping the breaker, or so the designers thought.

The pivots in the trip mechanism required lubrication that was not provided. The spring was not strong enough to overcome the lack of lubrication since the relay pulling on the spring was not forceful enough. After time, when the voltage holding the relay fell below its drop-out point, the spring was not strong enough to trip the breaker. Since this mechanism was the same in all the breakers, none of the breakers would trip on low voltage.[1]

BWRs Scram Mechanism Failures

A number of events involving common-mode failure in boiling water reactor (BWR) scram systems and their related support functions such as the air system and electric power system occurred. Specifically, it was discovered that water could be in the scram discharge volume (SDV) of a BWR as a result of poor drainage or an air supply failure. Water in the SDV would inhibit the insertion of control rods. The failure involving the air system was of particular concern because it involved a system that had been considered a portion of the reactor protection system not related to safety. Action was taken at all boiling-water reactors to correct this problem.[2]

This group of common-mode failures resulted from the use of a “fail-safe” design approach that in reality required a number of non-safety-related features to function and, therefore, the design did not truly rely on fail-safe principles. In the case of the air system, the system was assumed to fail safe, i.e., bleed off, and, as a result, a partial failure was not adequately addressed in the design. It was also noted that the electric supply system to this scram system had been modified previously because of a similar type of concern. Specifically, the electric power was originally assumed to fail safe (i.e., voltage going to zero) and, as a result, partial failure (such as low voltage or high voltage) was not appropriately addressed. In one case, 76 out of 185 “independent” control rods failed to fully insert.[3]

Scram discharge volume level monitoring systems designed to detect these failures also exhibited common mode failures themselves. In one plant operators discovered that the scram discharge volume (SDV)

continuous water level monitoring system had failed to respond as expected. The continuous water level utilized a single transmit/receive ultrasonic transducer for each of the four level monitoring channels.

The expected high level annunciator alarms were not received in the control room following scram when the SDV filled with water. Investigations determined that the ultrasonic detectors were inadequately coupled to the SDV piping.[4] This problem was compounded by noise problems created by cable routing, improper sensor placement, and cross-talk between detector channels.[5] In other cases, level sensor designs using differential pressure transmitters and resistance temperature detectors had unacceptably long response times.[6]

The HFA Relays

The HFA relays were commonly used in relay logic protection systems. They were widely used in the industry. In most applications they were continuously energized and operated in relatively tight enclosures that were designed to keep dust, dirt, and other industrial contaminants out of the relay mechanisms and contacts.

A series of problems were encountered that prevented these relays from changing state when de-energized. These problems included coil spool cracking,[7] melted coil spools,[8] and deposition of vaporized insulating materials on relay armatures or contacts.[9] These failures were common to many functions at some plants and occurred at a large number of plants.

A Danish Study

In 1975, J. R. Taylor of the Danish Atomic Energy Commission published a report entitled “Common Mode and Coupled Failures”[10] in which he studies event reports from the USA. He characterizes common-mode and common-cause events and classifies the events of the study according to their characteristics. He then proceeds to establish statistical models for common-mode and coupled failures.

How Have We Managed to Survive

To date, the conservatism of reactor designs together with D-in-D&D have kept the industry out of trouble. Defense-in-depth is the concept of multiple lines of defense against a perceived threat so that if one line of defense is penetrated, another line is invoked to limit the damage caused by the penetration. This can be carried through several levels. For example, to prevent the release of radioactive material to the environment, there are three physical barriers, each of which is supported by the three diverse control systems: the instrumentation and control (I&C) system of the reactor, the reactor protection system, and the Engineering Safety Features Actuation System (ESFAS). The first line of defense is the cladding on the fuel rods. This line is supported by (1) the reactor I&C system that attempts to keep the rector operating in the normal power producing regime, (2) the reactor protection system that will scram the reactor if required, and (3) the ESFAS system that will attempt to keep the reactor cool in emergency situations. If for some reason the various reactor cooling systems fail, the next physical line is the reactor pressure vessel that attempts to keep radioactive material from the failed fuel rods within the vessel. The vessel is also supported by the three control systems mentioned. Finally, the last boundary is the containment system which is designed to prevent release to the atmosphere if the pressure vessel fails. Containment is supported by the ESFAS. Lastly, the evacuation plan is invoked if all else fails. This is the classic D-in-D&D strategy and it has been successful in that evacuation plans have never been invoked.

Diversity is the notion that if redundant systems are different in some substantial way from each other, a failure in one will not necessarily imply a failure in the other. For example, some emergency core cooling pumps may be electrically driven, some steam driven, and the pumps themselves may come from different manufacturers.

The other phenomenon that has worked to our advantage has been alluded to previously. Because of variability in manufacturing, application environment, and usage, many CMFs look like random failures

since they occur randomly in time and the likelihood of all redundant systems failing simultaneously is small. But, if there is a common-mode mechanism at work, this likelihood is considerably larger than it would be otherwise, particularly if operational and environmental conditions are similar.

Software as a Particular Vulnerability

With the advent of digital computer-based systems and the attendant software, the problem of CMF has become more pressing. Software errors in one system are now carried exactly to the other redundant systems; no variability in manufacturing, application environment, or usage will produce any time randomness in the execution of the bug. Since each computer of the redundant set is seeing closely similar, if not identical, inputs at approximately the same time, the same software will be invoked in each computer. When a software error exists, all will execute a bug more-or-less simultaneously, and if this error causes the system to operate improperly or crash, each of the redundant machines will fail simultaneously.

N-Version Programming

As an answer to this problem, a method called "n-version programming" has been proposed. The notion here is that, working from the same software specification, several independent groups of programmers would write the code to meet the specification, perhaps even using different programming languages. This idea has not met with much acceptance for several reasons. First, the cost of maintaining several different sets of software has been daunting. More importantly, studies have shown[11] that the linkage at the requirements level is fatal. If the requirement is wrong, the software cannot be expected to overcome the problem. Further, there is some evidence that coding errors are not the big problem; errors in the requirements cause much more trouble than coding errors.

Software Testing

For large complex software systems, exhaustive testing becomes impossible because of time constraints. Typically a software vendor will send pre-release versions of software systems to friendly customers with instructions that they would like the customer to use the software in his normal course of work with the understanding that they are doing beta testing on the software. (Alpha testing is what the vendor does before he distributes it to beta testers.) The understanding includes a "hold harmless" clause and the customer agrees to report bugs to the vendor. After several months of beta testing, the vendor allows the product to go to paying customers and those customers are allowed to find the rest of the bugs. This is the reality. The number of software releases without bugs is small. Mostly it is hoped that the remaining bugs are no more than irritants.

In the reactor business, beta testing is not accepted. The software is expected to be free of bugs. Testing gets rid of most of the problems, but what about the remaining problems? Are there any problems remaining? The vendor would like to believe "no," but the reality is "yes." The issue is how to get the bugs down to an acceptable minimum and what to do about those that remain.

Software Quality Control

The ability of professional software production organizations to produce operating software with only benign bugs has greatly improved in the past decade. The reason for this is the introduction of "software engineering" techniques. The days when a customer loosely described what he wanted vocally to a coding hacker are mostly gone. Today, a Software Requirements Specification is developed and put under configuration control. Next, design proceeds in a stepwise evolution of requirements to conceptual design, detailed design, component design as the software process moves forward to produce code. (This is very similar to the traditional hardware design process.) This paper will not discuss the software engineering process in detail, but it has been very successful. The software engineering effort starts with a commitment by all the members of the software organization to high software quality and "bug free" operation. Currently, software flies our commercial airplanes, navigates our wartime missiles, runs our traffic lights,

and is included in many of our ordinary day-to-day activities. Software engineering works and is getting better all the time. But it is not perfect. So what do we do to guarantee the safety of the public in reactor operations?

First of all, do software bugs pose a threat to reactor safety?

What is the Experience?

Software has been implicated in several safety failures, one of which actually caused several deaths. Not all of the examples below are reactor examples, but they are illustrative of what can occur when software fails

Therac-25

The Therac-25 was a medical accelerator used for irradiating cancer patients. It was a two-mode machine, one mode being an x-ray mode and the other where the patient received an electron beam directly. The machine was controlled by a computer that took information from the operator and set the machine up for operation.

Unfortunately, the software was very overcomplicated for the application and had a mode that it got into if the operator typed too fast wherein it would do the wrong things. This resulted in several patients getting severe doses of radiation which, in several cases, killed.

This was written up in great detail by Nancy Leveson.[12]

The Space Shuttle

The space shuttle uses four redundant computers for control of the shuttle during launch and at other times. These machines are synchronized using a common clock. There is a fifth computer that serves as a backup to these four. It is a diverse machine running diverse software. It uses the common clock for synchronization.

During a launch sequence, while the vehicle was on the pad and within a few minutes of the launch, the four main computers appeared to fail. The launch was aborted and—other than a few red faces—there was no damage. However, if this failure had occurred during the launch, the mission would have aborted. In that case, a second problem involving an unexpectedly high loft pitch angle of the shuttle would have caused failure of the software designed to steer the shuttle to a landing. The results of these couple failures could have been fatal to the crew.

The problem was clock synchronization.

The people programming for the space shuttle pride themselves on writing bug-free code. They have an extensive QC program and have a great deal of pride in never having killed anyone. But considering the example, they were just lucky.

This is not a reactor example, but it illustrates what can occur despite all of the best efforts.[13]

Turkey Point Load Sequencer

This sequencer used multiple redundant programmable logic controllers (PLCs) to perform its function. The sequencer had self-test features which allowed individual PLCs to perform local tests to verify proper operation. In the event that the diesel generators were required while a unit was in self-test, the self-test mode was to be abandoned and normal operation was to be resumed. But there was a bug in the code and just when the sequencer was called upon, all of the units were in self-test. The bug prevented the units from resuming normal operation. As it turns out this problem was found during a testing operation so there were no safety consequences of this failure.[14]

South Texas Qualified Display Processing Software Error

An over-temperature/delta-temperature (OT/delta T) reactor trip occurred due to a software error. The reactor trip occurred at a reactor coolant system temperature of 530°F. Initial observation was that channel II T-hot output rose to 568°F giving a OT/delta T trip coincident with an OT/delta T trip already inserted in channel IV due to maintenance work. Subsequent investigation resulted in the identification of a software design error in the qualified display processing system (QDPS) temperature averaging system which erroneously calculated an average hot leg temperature of 568°F for channel II when the temperature was actually 530°F.[15]

The error was in the conservative direction and occurred prior to initial plant criticality. Therefore, there were no safety consequences. Similar errors in the non-conservative direction are certainly possible.

Additional Examples of Common-Mode Failure in Digital Systems

A U.S. Nuclear Regulatory Commission (NRC) study examined failures of digital I&C systems in U.S. nuclear power plants from 1990 through 1993. This study found that software errors was one of the prime causes of failures in these systems.[16] Each software error represents a potential common-mode failure.

The U.S. Nuclear Regulatory Commission Four-Point Position

The NRC debated for a considerable time on the problem of CMFs in redundant digital computer systems. The solution was the time-tested D-in-D&D that has been so effective over the years. The problem for software systems is, then, to establish what is adequate and effective for D-in-D&D for nuclear reactors in the U.S.

In order to establish the ground rules for what would be considered adequate D-in-D&D for computer based systems, the NRC has developed a four-point position that has been formally approved by the NRC.[17] The four points are:

1. The applicant shall assess the defense-in-depth and diversity of the proposed instrumentation and control system to demonstrate that vulnerabilities to common-mode failure have been adequately addressed.

2. In performing the assessment, the vendor or applicant shall analyze the postulated common-mode failure for each event that is evaluated in the accident analysis section of the safety analysis report (SAR) using best-estimate (using realistic assumptions) methods.

3. If a postulated common-mode failure could disable a safety function, then a diverse means, with a documented basis that the diverse means is unlikely to be subject to the same common-mode failure, shall be required to perform either the same function or a different function. The diverse or different function may be performed by a non-safety system if the system is of sufficient quality to perform the necessary function under the associated event conditions.

4. A set of displays and controls located in the main control room shall be provided for the manual system-level actuation of critical safety functions and monitoring or parameters that support the safety functions. The displays and controls shall be independent and diverse from the safety computer system identified in items 1 and 3 above.

Note that in point four, above, there is a requirement for "...the same function or a different function." If the "different function" is invoked, this paragraph is interpreted to mean that the different function has the same or equivalent system function as the "...same function." That is, if one method for cooling the core fails, another method may be started up.

D-in-D&D Assessment – NUREG/CR-6303

For a number of years the NRC relied on NUREG-0493[18] to provide a method and a model for D-in-D&D assessment. This NUREG was very effective for the purpose, but as computer systems and their developers got more sophisticated, the NUREG needed to be updated to account for this evolution. NUREG/CR-6303 is that update,[19] building on the solid foundation provided by NUREG-0493 and the experience gained by its use. Of course, as with all NUREGs, NUREG/CR-6303 is advisory and the method described is not mandatory. The assessment method in NUREG/CR-6303 was used in the assessment of all of the Advanced Light Water Reactor (ALWR) designs.

The Block Concept

To do a D-in-D&D assessment using the method outlined in the NUREG, the system needs to be divided into reasonably sized blocks and for each a reasonable probability that a failure in that block will cause all identical blocks to fail in the redundant systems. If these blocks are too big, the assessment is simple but it produces an assessment that is much too harsh. If the blocks are too small, the assessment becomes too complex to be accomplished in a reasonable time.

Once the blocks have been chosen, an analysis is performed for each block and each event in the catalog of events to which the system must respond effectively. It is presumed that all blocks similarly located in each of the redundant systems will fail at the same time for each event in the accident analysis. Then for each event with the chosen block inoperative, the remainder of the system is examined to see if it has enough diverse functionality left to respond effectively to the event.

Types of Diversity

If a block fails, the question becomes one of deciding whether or not the remaining system is sufficiently diverse from the failed block so that it will continue to function adequately in the face of the failure. It must be able to respond to the failure in such a way that the overall system is put or kept in a safe state.

There are a number of types of diversity outlined in the NUREG. Briefly, there is equipment diversity, functional diversity, human diversity, signal diversity, and software diversity. The NUREG explains why all diversities are not created equal and how it is decided whether or not adequate diversity exists. In particular, functional diversity and signal diversity are particularly strong forms of diversity. Decisions regarding what is diverse and what is not are not particularly objective and engineering judgment plays an important role in diversity assessment.

It is particularly important that the analysis be performed by highly competent engineering staff with a thorough and complete understanding of system performance. Otherwise the analysis may be worse than useless in that it will either give too optimistic an assessment of system performance or it will condemn a system that is otherwise perfectly usable. It should be noted that the former assessment is much more common than the latter so that analysis is typically not fail-safe.

Conclusion

Common-mode failures in hardware were not typically much of a problem in traditional reactor systems. With the introduction of digital systems into reactor protection systems, the possibility that the entire reactor protection system would fail at a critical juncture in reactor operations became much more plausible. This was attributed to the issue of common software being operated in all units of a redundant system where a bug in one would be a bug in all.

Testing to find these latent bugs in the software is not a reasonable means of coping with this eventuality because of the huge number of variations that must be tested. The time required for exhaustive testing is simply not available if a system is to be fielded in a reasonable time. Testing is an essential part of software production but experience shows that latent bugs are a fact of software production. However, the number of bugs in a code can be greatly reduced by good software engineering practice and a firm commitment to high software quality. The notion of “safety critical software” is now part of the software engineering world and methods are in place that allow the production of software for safety-critical applications. An example is the “fly-by-wire” systems in commercial aircraft.

The problem of common-mode failure in redundant systems can be addressed by defense-in-depth and diversity in the design of those systems. NUREG/CR-6303 provides a method for assessing D-in-D&D for digital systems which may have the potential for common-mode failure in redundant units running common software. The analysis method outlined in the NUREG has been successfully applied to several designs for nuclear reactor protection systems.

References

[1] NUREG-0977. “NRC Fact-Finding Task Force Report on the ATWS Event at Salem Nuclear Generating Station, Unit 1, February 22 and 25, 1983.” March 1983.

[2] Generic Letter 89-18. “Resolution of Unresolved Safety Issue A-17: ‘Systems Interactions In Nuclear Power Plants.’” September 6, 1989.

[3] IE Bulletin No. 80-17. “Failure of 76 of 185 Control Rods to Fully Insert During a Scram at a BWR.” July 3, 1980.

[4] IE Information Notice No. 80-43. “Failures of the Continuous Water Level Monitor for the Scram Discharge Volume at Dresden Unit No. 2.” December 5, 1980.

[5] IE Supplement 4 to Bulletin No. 80-17. “Failure of Control Rods to Insert During a Scram at a BWR.” December 18, 1980.

[6] IE Information Notice No. 87-17. “Response Time of Scram Instrument Volume Level Detectors.” April 7, 1987.

[7] IE Information Notice No. 81-01. “Possible Failures of General Electric Type HFA Relays.” January 16, 1981.

[8] IE Information Notice No. 82-13. “Failures of General Electric Type HFA Relays.” May 10, 1982.

[9] IE Bulletin No. 84-02. “Failures of General Electric Type HFA Relays in Use in Class 1E Safety Systems.” March 12, 1984.

[10] Taylor, J.R. “Common Mode Coupled Failures,” Danish Atomic Energy Commission, 1975.

[11] Knight, John C. and Nancy G. Leveson. “An Experimental Evaluation of the Assumption of Independence in Multiversion Programming.” In *IEEE Transactions on Software Engineering*, Vol. SE-12, no. 1, January 1986.

[12] Leveson, Nancy G. and Clark S. Turner. “An Investigation of the Therac-25 Accidents.” *Computer* (1993): 18-41.

[13] “Shuttle First Put to the Test,” *IEEE Spectrum* (August 1981).

[14] LER 94-005-02. “Turkey Point Units 3 And 4, Design Defect in Safeguards Bus Sequencer Test Logic Places Both Units Outside the Design Basis Event.” November 3, 1994.

[15] LER 88-014 REV 01. “South Texas 1, Update on Reactor Protection System Actuation due to a Software Problem in QDPS.” May 18, 1988.

[16] U.S. Nuclear Regulatory Commission. “Technical Review Report, Computer Based Digital System Failures.” AEOD/T94-03, July 1994.

[17] SECY-93-087. “Policy, Technical, and Licensing Issues Pertaining to Evolutionary and Advanced Light-Water Reactor (ALWR) Designs.” April 2, 1993.

[18] NUREG-0493. “A Defense-in-Depth & Diversity Assessment of the RESAR-414 Integrated Protection System.” March 1979.

[19] NUREG/CR-6303. “Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems.” December 1994.

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551