COMPUTER SYSTEM RELIABILITY AND NUCLEAR WAR

Given the devastating consequences of nuclear war, it is appropriate to look at current and planned uses of computers in nuclear weapons command and control systems, and to examine whether these systems can fulfill their intended roles.

ALAN BORNING

How dependent should society be on computer systems and computer decision making? What are the cost–benefit trade-offs between the advantages of computerization (greater efficiency, speed, precision, and so forth), and the jeopardy we are in when critical computer systems break down or otherwise fail to meet our intentions? These questions arise most compellingly in the use of computers in command and control systems for nuclear weapons, and it is on such uses that this article will concentrate. In this context the problem of defining "reliability" is clearly at issue. Obviously the concept extends beyond merely keeping a system running and invades the realm of system intention or even of *what we* should *have intended—had we only known.* To what extent are we able to state and codify our intentions in computer systems so that *all* circumstances are covered? Such questions have profound implications for the entire field of computer science; they also have important practical implications about how and where it is appropriate for us to use computers in critical systems.

Computers are used extensively in military applications: for managing data on friendly and enemy forces, simulating possible battles, and aiding in the design of weapons systems, as well as for such mundane tasks as keeping track of personnel, inventories, and payrolls. Nuclear forces in particular depend heavily on computer systems to guide missiles, analyze sensor data and warn of possible attack, and control communications systems. In fact, it would be quite impossible at present to do without computers in these systems. The short warning times required by current nuclear strategies, for example, necessitate the use of computers for data analysis and control of communications systems. Computers also play an essential role in the monitoring systems used to verify arms control agreements and could play an important role in a future crisis control center. Several aspects of nuclear weapons systems and strategy that interact in significant ways with computer system reliability are discussed here.

Portions of a preliminary version of this article were presented at the Fourth Congress of the International Physicians for the Prevention of Nuclear War and appeared in the *IPPNW Report*, volume 2, number 3, October 1984.

FALSE ALERTS

On several occasions, the North American Aerospace Defense Command Center (NORAD) early warning system has mistakenly indicated that Soviet missiles were headed for the United States. These incidents raise certain questions: Could a computer failure, in either the U.S. or the Soviet warning systems, start an accidental nuclear war? What risks are associated with placing the nuclear forces of one or both powers on alert? Would it be responsible for a country to adopt a policy of launch-on-warning, in which missiles would be fired based on warnings that an attack was imminent? Only the nuclear forces of the United States and the USSR are examined here, but the warning systems and nuclear forces of other countries clearly add to the problems described.

Missile Attack Warning Systems

Both the United States and the Soviet Union maintain elaborate systems for the detection of attack by enemy missiles presumed to be carrying nuclear

weapons. The primary sensors include satellites that can detect the infrared signature of a burning missile engine seconds after launch, and a variety of radar systems that can detect missiles in flight. Raw sensor data from the satellites must be processed by computer; processed data are available within minutes. Intercontinental ballistic missiles (ICBMs) have a flight time of about 30 minutes from the USSR to the United States; the U.S. ballistic missile early warning radars in Alaska, Greenland, and Britain can detect such missiles within 15 minutes, about halfway into their flight. Similar times apply to missiles launched from the United States toward the Soviet Union. There are shorter flight times for missiles launched from submarines off the coast of the enemy country, or for missiles launched from Western Europe toward the Soviet Union or vice versa [70, 120].

In the United States, the command post for the missile attack warning system is at NORAD, located 1200 feet under the solid granite of Cheyenne Mountain, Colorado. Other ground stations are located elsewhere in the United States and abroad. In the Soviet Union, the Air Defense Forces are responsible for early warning of nuclear attack and for attack assessment. A central underground Air Defense command center, similar to NORAD, is reportedly located about 50 kilometers from Moscow; there is also an extensive network of satellites, missile early warning radars, and communications facilities. (See [7] and [10] for more detailed descriptions of these systems.)

June 1980

On Tuesday, June 3, 1980, at 1:26 A.M., the display system at the command post of the Strategic Air Command (SAC) at Offutt Air Force Base near Omaha, Nebraska, indicated that two submarine-launched ballistic missiles (SLBMs) were headed toward the United States. Eighteen seconds later, the system showed an increased number of SLBM launches. SAC personnel called NORAD, who stated that they had no indication of SLBM launches. After a brief period, the SAC screens cleared. Shortly thereafter, the warning display at SAC indicated that Soviet ICBMs had been launched toward the United States. Then the display at the National Military Command Center (NMCC) in the Pentagon indicated that SLBMs had been launched. The SAC duty controller directed all alert crews to move to their B-52 bombers and to start their engines, so that the planes could take off quickly and not be destroyed on the ground by a nuclear attack. Land-based missile crews were put on a higher state of alert, and battle-control aircraft prepared for flight. In Hawaii, the airborne command post of the Pacific Command took off, ready to pass messages to U.S. warships if necessary. In the meantime, a Threat Assessment Conference was convened among the top duty officers at NORAD, SAC, and NMCC.[1] For the next three minutes, there was discussion among the three officers. There were a number of factors that made them doubt that an actual attack was under way: NORAD itself had no indications of an attack, the indications on the displays at SAC and NMCC did not follow any logical pattern, and the different command posts were receiving different information. Three minutes and 12 seconds into the alert, it was canceled. It was a false alert.

NORAD left the system in the same configuration in hopes that the error would repeat itself. The mistake recurred three days later, on June 6 at 3:38 P.M., with SAC again receiving indications of an ICBM attack. Again, SAC crews were sent to their aircraft and ordered to start their engines.

The cause of these incidents was eventually traced to the failure of a 74175 integrated circuit chip in a Data General computer used as a communications multiplexer. This machine took the results of analysis of sensor data and was part of the system that transmitted it from NORAD to SAC, NMCC, and Canadian Headquarters in Ottawa. The communications links were constantly tested by means of sending filler messages. At the time of the false alerts, these filler messages had the same form as attack messages, but with a zero filled in for the number of missiles detected. The system did not use any of the standard error correction or detection schemes for these messages. When the chip failed, the system started filling in the "missiles detected" field with random digits.

These false alerts received considerable press attention at the time [3, 29, 60, 139]. As a result of the publicity, on June 20, Senators Gary Hart and Barry Goldwater were asked to investigate the incidents by Senator John Stennis, chairman of the Senate Committee on Armed Services. They prepared both classified and unclassified versions of a report; the unclassified report [64] was the principal source of information for the above account of the incident. Other relevant U.S. government documents include [26], [34], and [132].

Other Incidents

The incidents of June 3 and 6, 1980, illustrate one sort of error—a hardware failure coupled with bad

[1] This is a formal step in the alert process. The successive levels of formal conferences are the Missile Display Conference, a relatively routine event; the Threat Assessment Conference, which is more serious; and the Missile Attack Conference, in which the president and all other senior personnel are brought in.

design—that can cause a false alert. Another incident illustrates a different realm of error: human operator error. On November 9, 1979, a test tape containing simulated attack data, used to test the missile warning system, was fed into a NORAD computer, which through human error was connected to the operational missile alert system. During the course of the ensuing six-minute alert, 10 tactical fighter aircraft were launched from bases in the northern United States and Canada, and as in the June 3 incident, a Threat Assessment Conference was convened [59, 64, 90]. (For information on other Threat Assessment Conferences during the period January 1977–May 1983 see [64] and the letter written by Col. J. H. Rix, director of administration at NORAD, to David C. Morrison, Center for Defense Information, Washington, D.C., November 4, 1983.)

Unsettling as the false alerts in November 1979 and June 1980 were, in the opinion of most reviewers of the incidents, including myself, the United States was nowhere near to launching its missiles and starting World War III. Most importantly, human judgment played an essential role in the procedures followed in the event of an alert, and these procedures provided enough time for the people involved to notice that a computer system was operating incorrectly. Also, NORAD procedures call for confirmation of the attack by an independent system—radar systems that observe the attacking missiles in flight, for example—and the chance of simultaneous false alerts from both systems under normal circumstances is very small.

What about similar failures in the Soviet warning systems? I have been unable to ascertain whether or not such failures have occurred, and it is unlikely that the Soviet government would choose to reveal them if they existed. For example, a recent paper on accidental nuclear war [46] by a member of the Soviet Academy of Sciences and former chairman of the State Committee for Atomic Energy of the USSR discusses U.S. warning system failures, without mentioning whether corresponding failures have occurred in the USSR. (Hints of the U.S. warning system failures were leaked to the press; the Pentagon stated that they would otherwise not have been made public [60].) At a news conference shortly after the June 1980 incident, Assistant Secretary of Defense Thomas Ross would not say whether the United States knew about similar false alerts in the USSR [60]. However, the Korean Airlines Flight 007 incident, in which a civilian aircraft was shot down by the USSR over two hours after it had entered Soviet airspace and just before it was back over international waters, would seem to indicate that the Soviet command and control system has problems.

We do know that the state of the art in Soviet computer science lags several years behind that in the United States [37, 56, 124]. However, the NORAD computers are very old by computing-industry standards,[2] whereas Soviet military computers are on the leading edge of their technology [127, p. 75].

TIGHTLY COUPLED NUCLEAR FORCES

In looking at the false alert of June 3, 1980, one is struck by the widespread effects of the failure of a single integrated circuit chip: Some 100 B-52 bomber crews were directed to start their engines, a battle-control aircraft took off in Hawaii, land-based missile crews were put on a higher state of alert, and submarines were notified. It is quite possible that some of these preparations were observed by the Soviet Union. After the incident, it was feared that such Soviet observations could in turn lead them to move their forces to a higher state of readiness, causing an escalating series of alerts and moving the two powers dangerously close to war.

Pentagon officials stated that in the case of the June 3 incident there was no discernible rise in the level of Soviet readiness in response to the U.S. alert. At a news conference shortly afterwards, however, when Assistant Secretary of Defense Ross was asked about this danger of escalating responses, his reply was, "I'm going to duck that question" [60]. Similarly, at a subsequent press conference, neither Assistant Secretary of Defense Gerald Dinneen nor other officials could assure that such a chain reaction would not be caused by another false alert [91]. The start of World War I following the assassination of the archduke in Sarajevo, in which the alerts and mobilizations of the European powers interacted in just this way, is a historical precedent for this possibility [21, 93, 137].

In response to the very short time in which a nuclear war could begin, the command and control systems of both the United States and the USSR have become highly reactive; given the possibilities of interacting alerts, we can view the nuclear weapons and control systems of both countries as a *single* interacting system. (This point is discussed at length in [21] and from a more mathematical viewpoint in [15].)

During times of relative international calm, the combined U.S.–Soviet system probably has enough human checks—more abstractly, enough stability or

[2] One 1982 congressional report termed the NORAD computers "dangerously obsolete" [26]. There have been upgrades since that report; nevertheless, the five largest on-line computers at NORAD are currently Honeywell 6080s (personal communication by D. W. Kindschi, chief of the Media Relations Division of NORAD, August 23, 1984). The machines in the Honeywell 6000 series were designed in the mid 1960s primarily for batch processing [126, p. 1]. Current plans call for replacing the Honeywell machines with IBM 3083 computers in 1988 [58, pp. 86–87].

hysteresis—to cope with a single mechanical or operator error, or perhaps even a few such errors. The situation would be different during a time of great tension or conventional war. Under such circumstances, the officers monitoring the systems would be less ready to dismiss a warning as being the result of a computer error, and the danger of escalating alerts on each side would be much greater. Again taking the single-system view, in times of tension and higher states of alert, the nuclear forces of the opposing sides become more tightly coupled.

A further danger comes from the possibility of compound stimuli to the system, perhaps from ambiguous or incomplete intelligence information. Bracken [21, pp. 65–66] describes one such example that occurred in 1956, at the time of the Suez Crisis and Hungarian uprising. On the night of November 5, the following four coincidental events occurred: First, U.S. military command headquarters in Europe received an urgent message that unidentified jet aircraft were flying over Turkey. Second, there were additional reports of 100 Soviet MiG-15 fighters over Syria. Third, there was a report that a British bomber had been shot down over Syria (presumably by the MiGs). Fourth, were reports that a Russian naval fleet was moving through the Dardanelles, perhaps to leave the Black Sea in preparation for hostilities. General Andrew Goodpaster was reportedly afraid that the events "might trigger off all the NATO operations plan," which at the time called for a single massive nuclear attack on the Soviet Union.

As it turned out, all four reports were incorrect or misinterpretations of more innocent activities: The jets over Turkey turned out to be a flock of swans, the MiGs over Syria were part of an official escort for the Syrian president, the British bomber was downed by mechanical difficulties, and the Russian fleet was on a scheduled exercise. In Bracken's words, "The detection and misinterpretation of these events, against the context of world tensions from Hungary and Suez, was the first major example of how the size and complexity of worldwide electronic warning systems could, at certain critical times, create a crisis momentum of its own."

The worldwide electronic warning and communications systems of today are immensely more complex and reactive than those of 1956. As in the 1956 incident, events that are in actuality unrelated may seem to be part of a larger pattern. Once the nuclear forces are placed on alert, further human or mechanical errors may occur. After the June 3, 1980, incident, the Hart–Goldwater report notes that, "Even though the command post controller prevented any undue reaction to the false and erroneous data, there seemed to be an air of confusion following the determination that the data were erroneous." It is likely that the "air of confusion" would be much worse if it were suspected that the indications of attack might be real.

An additional complication is the growing use of computer systems for "data fusion." One defense industry manager writes, "The most challenging information problem in modern command, control, communications and intelligence (C^3I) systems is the merging of diverse data into a single, coherent representation of the tactical, operational, or strategic situation. As C^3I systems have increased in complexity and scope, manual methods of merging data are no longer adequate, resulting in the need for fully automated methods, variously referred to as data fusion, multisource correlation or multisensor integration" [138, p. 217]. The motivation for this computerization is clear. The dangers are clear as well. As the amount of data increases and the time requirements become more stringent, less and less time is available for humans to check the outputs of the computer systems. Computer systems (including current artificial intelligence systems) are notoriously lacking in common sense: The system itself will typically not indicate that something has gone amiss and that the limits of its capabilities have been exceeded. This is an important aspect of automatic systems and one to which we will return.

To be at all confident about the reliability of complex systems, there must be a period of testing under conditions of actual use. As far as is publicly known, the command and control systems of the United States and the USSR have never been "tested" under conditions of simultaneous high alert; in fact, the

... on October 5, 1960, the warning system at NORAD indicated that the United States was under massive attack by Soviet missiles with a certainty of 99.9 percent. It turned out that the BMEWS radar in Thule, Greenland, had spotted the rising moon. *Nobody had thought about the moon when specifying how the system should act.*

highest level of conference in the U.S. missile warning system, the Missile Attack Conference, has never been called [64, p. 5]. Further, in a crisis situation, the very short times available for military personnel and national leaders to react and make decisions will undoubtedly lead to poorer judgment than under more usual circumstances, increasing the chances of misinterpretation of data and of error in operation of systems [18, 50]. The combination of the untestability of the warning and control systems under highly stressed conditions and the short times available for making decisions is grounds for considerable concern.

LAUNCH-ON-WARNING

Launch-on-warning is a strategy for retaliation to a nuclear attack, under which retaliatory missiles are launched in response to sensor indication that enemy missiles are on the way, before the warheads on the attacking missiles have detonated [52, 94, 109]. This strategy stands in contrast to "riding out the attack," a strategy in which a nation would absorb a nuclear strike and would retaliate only after positive verification had been obtained that an attack has taken place.[3]

Launch-on-warning makes stringent demands on a nation's nuclear weapons command and control systems. Warning data from sensors must be processed quickly, and it must be possible to relay launch orders through the command system quickly enough that missiles can be launched before the enemy missiles strike. Most importantly, the warning system must be exceedingly reliable, lest a retaliatory strike be triggered not by an enemy attack, but by computer or other error. (In recognition of this, if launch-on-warning were adopted as a strategy, it almost certainly would be activated only in times of crisis, rather than continuously [21, pp. 43–44]. Note that a policy of activation on this basis is an admission of distrust in the complete reliability of the warning systems and would result in a questionable system being activated at precisely the moment when the greatest caution was required.)

Given this danger of unintentional nuclear war, why would we consider adopting launch-on-warning? The reason is that the land-based missiles of both the United States and the Soviet Union have been growing more accurate over the years.[4] This increased missile accuracy puts at risk all fixed targets, such as land-based missile silos and command centers, even highly hardened ones. Although it is not at all certain that this vulnerability of fixed targets implies that a first strike could be successfully launched [28], strategic planners in both the United States and the USSR have nevertheless been concerned for decades with the problem. One way of dealing with it is launch-on-warning: If one side believes that an enemy attack is coming, retaliatory missiles can be launched and on their way, leaving the attacking warheads to explode on empty missile silos.

Although weapons based on submarines at sea and on aircraft are not threatened by this increasing accuracy, the present U.S. doctrine calls for all three "legs of the strategic triad" to be capable of inflicting retaliation. The risks to deterrence are more acute for the Soviet Union, which has a higher proportion of its strategic nuclear weapons on land-based missiles.

Launch-on-Warning Proposals in the United States

In April 1983, the President's Commission on Strategic Forces (often referred to as the Scowcroft Commission, after its chairman, retired Air Force Lt. Gen. Brent Scowcroft) issued its report on basing alternatives for the MX missile [109]. Acting on the committee's recommendation, the Reagan administration abandoned the goal of alternate basing modes and instead proposed that MX missiles be placed in existing Minuteman silos. This of course would leave them as vulnerable to Soviet attack as the Minuteman missiles.

In May 1983, in testimony before the Senate Appropriations Committee, Secretary of Defense Caspar Weinberger and the chairman of the Joint Chiefs of Staff, Gen. John Vessey, Jr., repeatedly told the committee that MX missiles deployed in existing silos would be vulnerable to a Soviet first strike "only if we ride out the attack" without launching a retaliatory strike. Vessey also said at one point, "The Soviets have no assurance that we will ride out the attack" [63]. However, on further questioning by senators, Weinberger and Vessey refused to say whether or not the United States was moving toward a launch-under-attack strategy.

Dr. Richard Garwin, a distinguished physicist and

[3] There are of course other strategies as well. In a launch-on-impact strategy, missiles are launched after indications have been received that at least one detonation has occurred. Launch-under-attack is defined differently by different authors: Sometimes it is used interchangeably with launch-on-warning, and sometimes to refer to a strategy that requires a higher confidence confirmation that an attack is under way. This higher confidence could be based either on reports of actual detonations (see [63]), making it the same as launch-on-impact, or on information from redundant sensors (as in [52]).

[4] For example, the U.S. Minuteman III Mk 12 missile has a reported accuracy of 280 m circular error probable, whereas the older Titan II missile has an accuracy of 1300 m. Similarly, the Soviet SS-18 Mod 3 missile has an accuracy of 350 m; the older SS-11 Mod 1 an accuracy of 1400 m [70, pp. 118–119]. The Pershing II missile is even more accurate. It uses a new guidance technology in which live radar images of the landscape surrounding the target area are compared during its descent with internally stored map information, so that course corrections can be made before impact. Its accuracy is reportedly 30 m [70, p. 118]. Missiles with similarly high accuracies are scheduled for deployment on nuclear submarines as well, for example, the D-5 missile due to be deployed on the U.S. Trident II submarines starting in 1989 [122, p. 54].

well-known defense consultant, has advocated the implementation of a system that can reliably support a launch-under-attack capability. The system would be enabled if it were determined that the U.S. submarine force had become vulnerable. In [52] Garwin advocates a system in which a limited number (50 or so) of Minutemen III missiles would be launched if an attack were detected. These missiles could be launched unarmed, subject to an encrypted command to arm them in flight. The decision process would be entirely predetermined, with the role of the U.S. National Command Authority (NCA) limited to assessing that a massive attack was indeed under way. Garwin also discusses alternatives, such as missiles launched armed but subject to a disarm command, or missiles launched irrevocably armed. In Garwin's proposal, an attack would be determined to be under way based on information from redundant infrared satellite sensors, not from reports of impacts or even radar data.

Launch-on-Warning Proposals in the Soviet Union

The Soviet Union has considered launch-on-warning as well, in particular as a threatened response to the Pershing II missile deployment by the NATO countries in Europe [41, 42, 140]. However, in a March 1983 statement, former Soviet Defense Minister Dmitri Ustinov categorically denied that the Soviets were adopting launch-on-warning [53]. While the political motivation behind these statements is clear, there appear to be real military issues as well. The Soviets would have 12 minutes or less from the time Pershing II missiles were launched until they hit [123, p. 46]. (Whether the current Pershing IIs could reach the command and control centers around Moscow is a matter of debate [122, p. 8].) The problem of short missile flight times is not new—missiles from Polaris submarines in the Arctic Ocean have been able to strike the Soviet Union since the 1960s—but the coupling of such short flight times with great accuracy *is* new.

More recently, Ustinov stated that the Soviet Union had increased the number of its nuclear-armed submarines off the U.S. coasts, to threaten the United States with more missiles with short flight times [110]. Although not the threatened response of launch-on-warning, in light of the previous discussion of tightly coupled forces, this action clearly has its dangers as well.

Discussion of Launch-on-Warning

Because of public perception of the risk of disaster due to computer or other error, the formal adoption of a launch-on-warning policy has always been controversial. Those authors who do advocate it do not appear to pay a great deal of attention to these dangers, particularly to the problems of very complex systems, short reaction times, and unanticipated events. For example, consider Garwin's discussion of accidental launch [52, pp. 124–125]:

> Launch under attack seems to present no more hazard of unauthorized or accidental launch than does the present system . . . .
>
> [The problem of an unauthorized launch] may be addressed by the use of PAL (permissive action links) in the silo and in the warhead. There are cryptographic safeguards which could be borrowed from modern message-security systems, which are adequate for the transmission of millions of characters per day with assurance against being "read" (deciphered) even if all the message traffic is intercepted by an enemy. These same systems could be used to encipher a short (20-digit) "go-code," receipt of which would cause the warhead to arm, while receipt of another go-code of similar length would fire the missile, having opened the silo door, and so on. The probability of accidental launch can be calculated as the number of candidate signals per year, times the likelihood that any one will be interpreted as a real go-code. Presumably very few putative go-codes would be received per year (the expected number is less than one per year, caused by lightning, electrical noise, or the like). If 1000 per minute were received, the pure-chance firing of the missiles would shorten the average human life by less than 0.1 seconds, even if only a single 20-digit code sufficed (and not 2, as assumed). For some cosmic-scale troublemaker to steal the actual go-code and so mimic the NCA launch order to the ICBM force, or bypass the wiring in the missile silos, is little different from what could be done now without a launch-under-attack system.
>
> . . . Only a limited number (say, 50) of Minutemen need or should be launched—unarmed, subject to command-arm in flight. One hopes that launch under attack will never occur inappropriately, in response to false indication of sensors, or other cause. Should such an unwarranted launch occur, however, we would prefer not to have armed the missiles, nor would we want to disarm ourselves by having launched the entire ICBM force, which would thus be lost to our future capability.

Some would argue that there are a number of important omissions in Garwin's analysis. The calculation of the probability of a randomly generated valid go-code, while correct in a narrow sense, is most misleading, as it ignores the host of other things that might go wrong. In Garwin's proposal, for example, the doctrine of dual phenomenology would be abandoned, so that a retaliatory strike would be launched based only on data from one kind of sensor, rather than two as at present. The discussion of launching missiles unarmed, subject to a command to arm them in flight, does not treat the real danger that the Soviet Union would observe the 50 missiles headed

The Strategic Defense Initiative (SDI) envisions a multilayer defense against nuclear ballistic missiles. The computer software to run such a defense would be the most complex ever built . . . it would be impossible to test the entire system under actual battle conditions short of fighting a nuclear war.

toward their territory and launch a retaliatory strike.

As mentioned previously, the formal adoption of a launch-on-warning policy—a declaration that launch-on-warning is the preferred response in a crisis—has always been controversial. Nevertheless, it appears that it is and has been regarded as an option by both the United States and the USSR. According to testimony by General Ellis of the U.S. Strategic Air Command [135, p. 3834],

> launch on warning is an option we have and must maintain. It remains a useful option because the enemy cannot be certain it will not be used or know the conditions under which it would be used . . . and therefore, he must always make it a part of his planning deliberations.

This is corroborated by recent testimony by General Herres, commander in chief, NORAD [134, p. 72]. A recent book on the U.S. Single Integrated Operating Plan (SIOP) for waging nuclear war states that launch-on-warning has *always* been an option in the SIOP [102, pp. 187–188]. More alarmingly, Bruce Blair, a former launch control officer and DoD official, has testified [134, pp. 32–34]

> declaratory doctrine is a poor guide to actual employment doctrine. At present, we are operationally geared for launch on warning, a reflection of the low confidence we have in our ability to absorb the brunt of an attack before retaliating. . . . I restress the fact that the United States relies heavily on launch on warning for positive control, for force coordination and for retaliation. Fortunately, our tactical warning system on which launch on warning depends is fairly fault tolerant. But again, it is not as tolerant as it should be to justify U.S. reliance on it.

Launch-on-warning is the subject of a current lawsuit,[5] in which the plantiff complains that the secretary of defense is presently operating a launch-on-warning capability. This operation, according to the suit, unconstitutionally usurps the power of Congress to declare war, and unlawfully delegates presidential powers to subordinates, since the very short times involved would not allow time for a decision by the president.

[5] Johnson v. Secretary of Defense, U.S. District Court, San Francisco, Calif., case C86 3334.

Regarding the Soviet Union, a DoD publication [129, p. 20] states

> launch-under-attack circumstances would place the greatest stress on attack warning systems and launch coordination. To meet this demand, the Soviets have established a satellite-based ICBM launch detection system, built an over-the-horizon radar missile launch detection system to back up the satellites and have large phased-array radars ringing the U.S.S.R. These warning devices could give the Soviet leadership time to launch their forces after an enemy strike had been launched. To prepare for this possibility, the Soviets practice launching weapons under stringent time constraints.

Because of the very short times involved, there is doubt that launch-on-warning is a practical policy [120], since it would be difficult to maintain an acceptable level of control on the nuclear forces of the country that adopted it. From a broader viewpoint, launch-on-warning can be seen as one point on a spectrum of policies for retaliation, the dimension of the spectrum being how long a country waits to respond when it believes that an attack is imminent or under way. Taking this broader view, pressures toward launch-on-warning are actually a symptom of underlying problems. Among these problems are (1) the strategic doctrine that holds that military assets at known, fixed locations (land-based ICBMs and command posts) are an essential part of a nation's nuclear forces, (2) the perception that the vulnerability of these fixed targets is a pressing problem, (3) new weapons systems that make them more vulnerable, and (4) the consequent decrease in time available to make decisions in nuclear crises. (See [120] for a longer treatment of this viewpoint.)

THE RELIABILITY OF COMPLEX SYSTEMS

Would it be responsible for either the USSR or the United States to adopt weapons systems and policies that assume that computer systems, such as missile warning systems, can function without failure?
I argue that it is not. I will not attempt to prove that failures will occur in complex military systems, but rather I will attempt to show that there is considerable doubt that adequate reliability can be achieved. The standard of reliability required of a military system that can potentially help precipitate a thermo-

nuclear war if it fails must be higher than that of *any* other computer system, since the magnitude of disaster is so great.

Techniques for Building Reliable Systems

Much research and development effort has been devoted to the construction of reliable computer systems, and some impressive results have been achieved. As a comprehensive treatment of this topic is well beyond the scope of this article, an outline of some well-known techniques for achieving reliability is presented, along with references to the literature, with particular emphasis on military computer systems.

Hardware. At the hardware level, one obvious technique is to use very reliable components. Here the large body of knowledge about quality control for other kinds of manufacturing can be applied, including quality control of raw materials, testing and tracking each component produced, destructively testing a certain percentage of the devices, and keeping records of the reliability of components produced by a particular line to spot variations in reliability. The MIL-SPEC program codifies standards for many kinds of devices that the military procures. In addition, the DoD has funded much work on building models of component reliability, such as the widely used MIL-HDBK-217C reliability model for estimating the failure rate for various kinds of integrated circuit chips [125]. Above the chip level, techniques for building reliable devices include component burn-in, careful signal routing, shielding, cabinet grounding, environmental controls, power supply regulation, and other conservative, well-established design practices.

Regardless of the methods used, in a very large system it is unreasonable to expect that every component will be totally reliable. For this reason, a body of techniques has been accumulated that allow a system to continue functioning even when individual components fail. These techniques all involve redundancy, and include n-modular redundancy with voting, error-correcting codes, and dynamically reconfigurable systems.

Complementing this work on the construction of reliable hardware has been development of modeling techniques; useful measures include mean time to failure, mean time to detection, mean time to repair, and availability. More information on hardware reliability, along with an extensive bibliography, may be found in [111]; [31] is a review of techniques for achieving hardware fault tolerance.

Software. For large computer systems, the cost and complexity of the software typically dominate that of the hardware. To construct a very complex system at all, let alone to make it reliable, a disciplined approach is necessary. An extensive set of sources discussing the software development process is available. Texts on software engineering include [19], [47], [71], and [115]; these have references to many other sources, including seminal papers on software engineering in the literature.

It is generally accepted that reliability cannot be "tested into" a software system; it is necessary to plan for reliability at all points in the development process. As with hardware, the DoD has codified standards for how its software is to be specified, designed, written, and tested. One such standard is *DOD-STD-2167: Military Standard Defense System Software Development* [131], for the development of mission-critical software. It specifies such things as software requirements analysis standards, coding standards, and the information that must be gathered on software trouble reports. In addition, other administrative requirements may be imposed, such as formal requirements for a contractor's Quality Assurance Program (MIL-STD-1535) and requirements for configuration control (DOD-STD-480). Formal reviews of the software development process are required at each stage by DoD directives. As enumerated in [2, p. 186], these are

- a *Systems Requirements Review*,
- a *System Design Review*,
- a *Preliminary Design Review*,
- a *Critical Design Review*,
- a *Functional Configuration Audit*,
- a *Physical Configuration Audit*, and
- a *Formal Qualification Review*.

The DoD will typically contract with a company (other than the software contractor) to assist it with some of these reviews.

Testing is not simply performed at the end of coding, but rather must be planned and developed in parallel with the software system itself. A typical contractual requirement would be that, for every item in the system specification, a corresponding test be performed to check that the software meets each specification. Even after the system is installed, the set of tests should be kept and updated as well, so that, as the system is modified during maintenance, previous tests can be rerun to check that the system still meets them (regression testing). More information on software reliability, safety, quality assurance, testing, and validation may be found in [2], [5], [40], [68], [77], [81], and [82].

Quantifying Software Reliability. Two kinds of software quality measures are in general use: estimates of the number of errors remaining in a program, and

estimates of the mean time to failure (MTTF) of a system. Angus [6] describes the application of six models of the first sort to a major C^3I system, with poor results. Regarding estimates of MTTF, Currit, Dyer, and Mills [36] describe a procedure for producing a certified estimate of the MTTF of a system, using statistical testing.

Any estimate of the errors remaining in a program requires a complete specification against which the program can be compared. The testing regimes assume either that the testers know what kinds of inputs the system will be subjected to, or that the system can be extensively tested under conditions of actual use. (Even then, a problem with statistical testing is that it takes prohibitively long to obtain high confidence that the errors found are manifested only rarely.) The importance of these limitations will be discussed later in the article.

Sources of System Failure

The sources of computer system failure include incorrect or incomplete system specifications, hardware failure, hardware design errors, software coding errors, software design errors, and human error (such as incorrect equipment operation or maintenance). Particularly with complex, normally highly reliable systems, a failure may be caused by some unusual *combination* of problems from several of these categories.

Hardware failures are perhaps the most familiar cause of system failures, as in the NORAD failures of June 1980. As noted previously, individual components can be made very reliable by strict quality control and testing, but in a large system it is unreasonable to expect that no component will ever fail, and other techniques that allow for individual component failures must be used. However, when one builds very complex systems—and a command and control system in its entirety is certainly an example of a complex system—one becomes less certain that one has anticipated all the possible failure modes, that all the assumptions about independence are correct. A serious complicating factor is that the redundancy techniques that allow for individual component failures themselves add additional complexity and possible sources of error to the system.

Another potential cause of failure is a hardware design error. Again, the main source of problems is not the operation of the system under the usual, expected set of events, but its operation when *unexpected* events occur. For example, timing problems due to an unanticipated set of asynchronous parallel events that seldom occurs are particularly hard to find.

It is in the nature of computer systems that much of the system design is embodied in the computer's software. Errors may be introduced at any of the steps in its production: requirements specification, design, implementation, testing and debugging, or maintenance.

Errors in the system requirements specification are perhaps the most pernicious. It is at this level that the system's connection with the outside world is expressed; we must therefore anticipate *all* the circumstances under which the system might be used and describe in the requirements specification what action it should take under those circumstances. For a very complex system, it is unrealistic to imagine that one can foresee all of these circumstances. We can have confidence in such systems only after they have been tested for a considerable time under conditions of actual use.

Errors may also be introduced when the requirements are translated into a system design, as well as when the design is translated into an actual computer program. Again, the sheer complexity of the system is itself a basic cause of problems. Anyone who has worked on a large computer system knows how difficult it is to manage the development process; usually, there is *nobody* who understands the entire system completely. Given a complete requirements document, however, many of the errors at these levels can be prevented by using strategies such as modular design, information hiding, and the like; also, a wider range of automated tools is available to help us detect which parts of the program affect which other parts. Nevertheless, it is widely acknowledged that the process is not completely satisfactory.

The cost of maintenance usually dominates the other costs of military software development. Program maintenance, either to fix bugs or to satisfy new system requirements, is itself a frequent source of errors. Meyers [81, p. 252], for instance, states that "experience has shown that fixes have a high probability (usually from 20 to 50 percent) of introducing a *new* error into the program."

Another source of failure is human operator error. People do make mistakes, despite elaborate training and precautions, especially in time of stress and crisis. Dumas [44] cites some worrying statistics about alcohol, drug abuse, and aberrant behavior among military personnel with access to nuclear weapons. Alcoholism is a major health problem in the Soviet Union and may be a problem among such personnel in the Soviet military as well [32].

Some Instructive Failures

There have been some impressive failures of computer (and other) systems designed to be reliable,

and it is instructive to look at a few of these. I have attempted to categorize these failures using the sources of failure listed in the previous section; however, as will be seen, these failures often arise from a combination of errors.

Examples of failures due to hardware errors include the NORAD false alerts described earlier. (However, it could also be said that these false alerts are illustrations of hardware design errors instead, in that it is a grave oversight that such critical data should have been sent without using parity, cyclic redundancy, or other checks.) From a technical point of view, a more interesting and complex failure was the total collapse of a U.S. computer communications network (the ARPANET) in October 1980 due to an unusual hardware malfunction that caused a high-priority process to run wild and devour resources needed by other processes [107]. The ARPANET was designed to be highly available—the intent of the software design was that it should prevent a single hardware malfunction from being able to bring down the whole network. It was only after several years of operation that this problem manifested itself.

The launch of the first space shuttle was delayed at the last minute by a software problem. For reliability, the shuttle used four redundant primary avionics computers, each running the same software, along with a fifth backup computer running a different system. In the incident, a patch to correct a previous timing bug opened a 1 in 67 probability window that, when the system was turned on, the computers would not be properly synchronized. There are a number of noteworthy features of this incident: First, despite great attention to reliability in the shuttle avionics, there was still a software failure; second, this particular problem arose from the additional complexity introduced by the redundant systems designed to achieve reliability; and third, the bug was introduced during maintenance to fix a previous problem. Garman [51] gives a detailed account of the incident, along with some pithy observations on the problems of complex software systems in the real world.

There are many examples of errors arising from incorrect or incomplete specifications. One such example is a false alert in the early days of the nuclear age [16, 69, 89], when on October 5, 1960, the warning system at NORAD indicated that the United States was under massive attack by Soviet missiles with a certainty of 99.9 percent. It turned out that the Ballistic Missile Early Warning System (BMEWS) radar in Thule, Greenland, had spotted the *rising moon.* Nobody had thought about the moon when specifying how the system should act.

Gemini V splashed down 100 miles from its intended landing point because a programmer had implicitly ignored the motion of the earth around the sun—in other words, had used an incorrect model [49, pp. 187–188]. In 1979 five nuclear reactors were shut down after the discovery of an error in the program used to predict how well the reactors would survive in earthquakes [87]. One subroutine, instead of taking the sum of the absolute values of a set of numbers, took their arithmetic sum instead.[6] In 1983 severe flooding along the lower Colorado River killed six persons and caused millions of dollars in damage. The governor of Nevada stated that this was caused by a "monumental mistake" in federal computer projections of snow melt-off flow, so that too much water was kept dammed prior to spring thaws [92].

ACM SIGSOFT Software Engineering Notes is a good place to find descriptions of real-world computer problems, catastrophic and otherwise (e.g., see [88]). See also [102] for a listing of some other incidents.

In hindsight, the blame for each of the above incidents can be assigned to individual component failures, faulty design, or specific human errors, as is almost always the case with such incidents. In designing automatic systems, we must anticipate all possible eventualities and specify what should happen in all cases. The real culprit is simply the complexity of the systems, and our inability to anticipate and plan for all of the things that can go wrong.

Outside of the realm of computer systems, incidents such as the tragic explosion of the space shuttle *Challenger* in 1986, the accidents at the nuclear power plants at Chernobyl in 1986 and Three Mile Island (TMI) in 1979, and the 1965 northeast power blackout are sobering reminders of the limitations of technology.

At Chernobyl, operators deliberately disabled warning and safety mechanisms so that they could conduct an experiment, with the catastrophic result of two explosions at the plant and the release of enormous amounts of radioactive material. The TMI accident began with an equipment failure (of a pressurizer relief valve), but its severity was compounded by subsequent operator error [106].[7] In another nuclear reactor accident, at Browns Ferry in Alabama in 1975, a single failure—a fire in an electrical cable tray—disabled a large number of redun-

[6] It is not clear whether this should be classified as an error in the specification or in the program—probably there *was* no separate formal specification or model, so that the program itself became the model.

[7] See also [100] and [101] for a discussion of the TMI incident as a "normal accident"—an unanticipated accident in a complex, tightly coupled system. This "normal accident" viewpoint is also applicable to other complex systems such as nuclear weapons command and control systems.

dant systems designed to ensure safety at the plant. This incident demonstrates that one should look with a skeptical eye at calculations indicating extremely low probabilities for failure due to independent systems.

Prospects for Military Computer System Reliability

What are the prospects for the reliability of military computer systems in the future?

Clearly, substantial improvements in the reliability of systems like NORAD are possible simply by using state-of-the-art hardware and software engineering techniques: A system that uses 1960s vintage computers or that as recently as 1980 transmitted critical data with no parity checks is not state of the art. Nor are these isolated incidents. The World-Wide Military Command and Control System (WWMCCS) has been plagued with problems of inadequate performance, cost overruns, and poor management [22]. In the 1977 PRIME TARGET exercises, for example, only 38 percent of the attempts to use the system were successful [33, p. 51]. There are also problems with personnel training and preparation. A recent book by Daniel Ford [48, p. 21] describes an incident in which Ford asked Gen. Paul Wagoner, at the time in charge of NORAD combat operations, to demonstrate the special black telephone that provides a direct link to the NMCC. This telephone would be used, for instance, for a Missile Attack Conference. Wagoner picked up the phone—and nothing happened. His subsequent explanation was that, "I didn't know that I had to dial '0' to get the operator." (See [17] for further discussion of the deficiencies of the current U.S. command and control system. Kling [75] discusses WWMCCS as an example of a socially complex computer system, points out the inadequacies of describing such systems in isolation, and advocates the use of "web models" as an appropriate tool for describing such systems.)

This is not to say that there have been *no* improvements in these systems—a good example of a positive step has been the installation of Permissive Action Links (PALs) on all U.S. nuclear weapons except SLBMs [83, p. 52]. The PAL system requires that a code be received from a higher authority before a nuclear weapon can be armed, thus reducing the probability of unauthorized use.

Nevertheless, substantial improvements are possible using existing state-of-the-art technology. What are the practical and theoretical limits of reliability, now and in the next decade?

In regard to the practical limits of reliability, most professional programmers today do not use such software engineering techniques as structured programming, modularity and information hiding, cooperating sequential processes, or formal program semantics [96]. The DoD is engaged in several efforts to develop new technology for software production and to make it widely available to military contractors [78]. The STARS (Software Technology for Adaptable, Reliable Systems) program [43, 128] and the Software Engineering Institute at Carnegie-Mellon University are examples. Large organizations move slowly, and it will be some years (at least) before these newer software engineering techniques are generally adopted. Use of these techniques should decrease, but not eliminate, errors in moving from the specification to the program.

Program maintenance, as noted previously, is itself a frequent source of errors. This problem is further aggravated by the fact that program maintenance is presently regarded as one of the least desirable programming jobs and is often assigned to junior or less-skilled employees. Programming support environments that keep track of versions, note the effects of changes, and the like are becoming available [12, 66]. Eventually, the use of these tools should help decrease the number of errors introduced during maintenance.

In the long term, formal techniques such as proofs of program correctness (program verification), automatic programming, and proofs of design consistency have been advocated as tools for improving computer system reliability. In a proof of program correctness, either a human or a computer proves mathematically that a program meets a formal specification of what it should do. In automatic programming, the program is written automatically from the specification. In a proof of design consistency, the proof must show that a formal specification satisfies a set of requirements, for example, for security or fault tolerance. (The difference between requirements and specifications in this case is generally that the former tend to be simply stated global properties, whereas the latter tend to be detailed sets of constraints defined functionally on state transitions or algebraically on inputs and outputs.)

In theory, these techniques could produce programs or designs guaranteed to meet their specifications. Some practical use is being made of design proof techniques, primarily in proving security properties of system designs, although such proofs are still nontrivial. Thus, one might prove that, within the computer, information cannot flow in the wrong direction in a multilevel security system.[8] Proofs about program correctness, however, are very much in the research stage. For example, simple compilers

[8] Design proofs have been done successfully in the SCOMP kernel [112], while other relevant properties have been proved about the trusted code that runs on top of the kernel [14].

have been proved correct, but programs of the complexity of the real-time satellite data analysis programs are well beyond the state of the art. Automatic programming is even less advanced. A useful reference discussing program verification is [20]; for an up-to-date collection of papers on verification, see [8]. The current state of automatic programming is discussed in [99]; discussions of future applications of automatic programming to software engineering may be found in [11] and [13]. In [97] Parnas critiques the possible roles that automatic programming and program verification could play in the production of software for ballistic missile defense.

The hardest and most intractable problem in the construction of software for complex tasks, such as command and control systems, is specifying what the system should do. *How does one know that the specification itself is correct, that is, that it describes what one intends?* Are there events that may occur that were simply not anticipated when the specification was written? Program verification and automatic programming techniques can offer no help here. A proof of correctness, for example, simply shows that one formal description (the specification) is equivalent to another formal description (the program). It does not say that the specification meets the perhaps unarticulated desires of the user, nor does it say anything about how well the system will perform in situations never imagined when the specification was written. For example, in the 1960 false alert, proving that the system met its specifications would not help if nobody thought about the rising moon when writing the specifications. (The term *proof of correctness* is thus a misnomer—a better term might be *proof of relative consistency.* This point is discussed at length in [113].)

Both the practical and theoretical limits of reliability bump up against this problem of specification. It constitutes the major long-term practical barrier to constructing reliable complex systems. From a theoretical point of view, depending on the language used to express the specification, it may be possible to prove that it has certain properties, for example, that it is self-consistent or that, given a set of possible inputs, the action to be taken for each of these inputs is specified. However, the answers to such critical questions as, "Will the system do what we reasonably expect it to do?" or "Are there external events that we just didn't think of?" lie inherently outside the realm of formal systems.

On Testing

To be at all confident of the reliability of complex systems, there must be a period of testing under conditions of actual use. Simulations, analyses of possible modes of failure, and the like can each expose some problems, but all such tests are limited by the fact that the designers test for exactly those circumstances that they anticipate may occur.[9] It is the *unexpected* circumstances and interactions that cause the most severe problems.

Some problems, like spotting the rising moon, will be uncovered quickly when the system is in routine operation. However, the conditions under which command and control systems for nuclear forces are expected to function include not just peacetime, but also times of international tension and high alert. It is these latter situations that are of the most concern. Short of having many periods of great tension and high alert—clearly an unacceptably dangerous proposition—the nuclear weapons command and control systems simply cannot be tested completely. The most extreme situation in which, under current doctrine, these systems are expected to function is that of limited or protracted nuclear war; this topic is discussed in the next section.

A final issue is that systems in flux are more prone to problems than those that have remained stable for some time. As noted above, program maintenance is a frequent source of errors. Better programming environments will help eliminate some of these errors, but such errors also arise from changing specifications, in which some loophole or problem in the specification is introduced by other changes. If the arms race continues unabated, due to the changing nature of the weapons and their deployment, the specifications for the command and control systems for nuclear forces will necessarily be changing as well.

LIMITED OR PROTRACTED NUCLEAR WAR

In this section a number of issues concerning limited or protracted nuclear war are examined. Many of the technical issues that arise concern the physical survival of computers and communications lines; there are also implications for the software that must run in such an environment.

Nuclear Strategy

Deterrence theory states that, to prevent nuclear war, each opponent must have nuclear weapons systems, strategies for using them, and the perceived will to retaliate in the event that deterrence fails [85]. One policy to implement deterrence is "mutual assured destruction" (MAD): An attack by one side would result in massive retaliation by the other, essentially resulting in mutual suicide.

Many strategists have long been dissatisfied with

[9] See [121] for an interesting although dated discussion of how the NORAD system was tested, including simulated battle exercises.

MAD. A fundamental problem is that, if the Soviet Union commits a very aggressive act that is short of an all-out attack, the president's only options are to do nothing or to launch an all-out attack (which would doubtless lead the Soviets to do the same). For this reason, there has been an evolution toward plans that include more flexible options (selective targeting of enemy nuclear forces, conventional forces, military and political leadership, communications facilities, industrial targets, and cities), and capabilities to fight limited or protracted nuclear wars. For example, directives issued under the Carter administration (such as Presidential Directive 59) called for provision of a wide variety of responses following a nuclear attack; a major change was the requirement to support fighting a protracted nuclear war, lasting perhaps months rather than days [7, pp. 459–460]. A more recent document from the Reagan administration, "Fiscal Year 1984–1988 Defense Guidance," which was leaked to the press in May 1982 [61], goes beyond PD-59 in recommendations for preparing for a protracted nuclear war. The document states that "the United States nuclear capabilities must prevail even under the conditions of a prolonged nuclear war" and that U.S. nuclear forces "must prevail and be able to force the Soviet Union to seek earliest termination of hostilities on terms favorable to the United States" [62].

According to a basic Soviet text, *Soviet Military Strategy* by Marshal V. D. Sokolovskiy [114, p. 279], "Apparently, in a nuclear war a victory can be counted upon only if the basic power is used in the shortest possible period. . . . At the same time, the possibility of a relatively protracted nuclear war cannot be excluded." Similarly, a DoD publication [129, p. 20] states that "the Soviets appear to believe that nuclear war might last for weeks, even months, and have factored this into their force development." Testimony before the Armed Services Committee, U.S. Senate [136, p. 2491], supports the view that both the United States and the USSR are building systems to support fighting a limited nuclear war.

These moves toward a capability to fight limited or protracted nuclear wars have been controversial. As described above, some strategists argue that, to maintain an effective deterrent, we must plan for nuclear conflict at any level of violence, and that we must plan for ways to control such a conflict if it breaks out at a level below an all-out attack [84, pp. 94–97]. Others maintain that, although some flexibility in response is essential, the sorts of carefully controlled responses now being planned deceive us into believing that nuclear war *can* be successfully controlled [84, pp. 130–131]. Taking this latter viewpoint, if leaders believe that nuclear war can be controlled, then, for example, in a severe crisis they may be less reluctant to launch a small tactical strike, thus making nuclear war more likely. More extensive discussions of deterrence and nuclear war strategies—and a variety of viewpoints—can be found in [54], [57], [67], [72], [73], [74], and [85].

Command, Control, and Communications System Requirements

Different strategies make different demands on a nation's command, control, and communications (C^3) system. Listed below are some basic attributes of a C^3 system that are governed by the choice of strategy:

- the length of time the system needs to survive during and after nuclear attacks of various scales,
- the amount of information that needs to be transmitted from the NCA to the nuclear forces,
- the amount of information that needs to be transmitted from the field back to the NCA, and
- the facilities for communicating with the enemy during and after the war.

How long does the C^3 system need to survive during and after nuclear attacks of various scales? The problem of C^3 vulnerability, analogous to the problem of land-based missile vulnerability, has begun to be widely discussed. One extreme position would be that the C^3 does not need to survive an attack at all: Retaliatory missiles would be launched on warning, obviating the need for both survivable missiles and communications systems. As previously discussed, such a policy would be quite dangerous. Blair [17, pp. 289–295] proposes as a long-term goal for the United States a quite different policy: "no immediate second use." Under such a policy, authority to conduct offensive operations would be withheld for 24 hours after a Soviet attack. Such a policy would lead to much greater stability in a crisis; the obvious question is whether the U.S. C^3 system could survive that long to permit an order to retaliate to be issued. (Survival of the weapons themselves is less of a problem, since submarine-based missiles have the desired characteristics.)

Another issue is how much information needs to be transmitted from the NCA to the nuclear forces, and how much information needs to be gathered from military installations and possible civilian targets and transmitted back to the NCA. Under a policy of MAD, a minimal amount of information needs to be transmitted from the NCA to the nuclear forces: 1 bit, along with authentication codes. No information need be transmitted in the other direction.

However, under policies that include flexible responses to various kinds of attacks, greater commu-

nications bandwidth from the NCA to the fighting units is required; further, information must move in the other direction as well, so that the NCA can receive damage reports, current warning information, and intelligence reports to use in deciding further responses. Preparing to fight a protracted nuclear war makes even greater demands on C^3 systems, which would have to survive for days or months through a nuclear war. The "Fiscal Year 1984–1988 Defense Guidance" document, for example, calls for C^3 systems that "provide the capability to execute ad hoc plans, even subsequent to repeated attacks . . . in particular, these systems should support the reconstitution and execution of strategic reserve forces, specifically full communications with our strategic submarines." A recent report by the U.S. secretary of defense to the Congress [141, p. 195] tends to confirm these plans:

> Our C^3 systems must be able to provide our leaders the information they need to assess the size and scope of an attack, determine an appropriate response, and issue initial retaliatory orders. These systems also must be able to ensure that our forces would receive those orders, called emergency action messages (EAMs), and remain responsive to national authority both during and after an attack.
>
> Strategic C^3 systems must be able to operate reliably under the extremely stressful conditions of a nuclear conflict. . . . The FY 1985–89 program will improve our strategic C^3 systems—sensors, command centers, and communications—by upgrading and augmenting their capabilities, increasing their mobility, protecting essential equipment against nuclear effects, and providing alternate and redundant methods of communication.

The nature of the facilities for communicating with the enemy during and after the war is also at issue. If a limited nuclear war strategy is to be pursued, a means to communicate with the enemy to call a cease-fire or to terminate the conflict is important. Theorists of nuclear war have described scenarios in which there would be a kind of communication by attacking or holding back attacks on given targets, but it is not clear that the signals would be interpreted correctly, or that in the emotion and tension of the war, the leaders of each country would react with the calm rationality assumed by these scenarios. Further, such theories require that the facilities for receiving damage reports and sending out new commands to the fighting units work extremely reliably during the conflict.

Prospects for C^3 System Reliability during a Nuclear War

Both the strategies of delayed and of flexible response require C^3 systems that can survive for some period in a nuclear conflict. Even if C^3 systems were built as well as we knew how, this requirement would be difficult to meet. Blasts destroy control centers and communications apparatus. Particularly when detonated high in the atmosphere, thermonuclear warheads can create an electromagnetic pulse (EMP), a strong electric field (up to 50,000 volts per meter) that can cripple computer equipment, communications and power lines, and other electrical and electronic apparatus [17, 23–25, 55, 118]. For example, after a 1962 test in the South Pacific, 800 miles away in Hawaii streetlights failed, burglar alarms started ringing, and circuit breakers opened. Modern integrated circuits are much more sensitive to these effects than are door bells and circuit breakers. Since the 1963 Limited Test Ban Treaty, which stopped atmospheric testing, data on EMP has been derived from EMP simulators, computer simulations, and underground weapons tests (obviously a different environment from a high-altitude airburst). Also, the electronic devices of today have changed greatly since 1963.

It is possible to harden electronic apparatus against EMP using a variety of techniques [39, 104, 105], for example, by enclosing it in a Faraday cage (a metal shield). (The wires to the outside world, which must pierce the cage, are a harder problem.) Another and less attractive technique is to use less sophisticated technologies, which are in general more resistant to EMP effects—for example, vacuum tubes rather than integrated circuits. Some systems, such as long transmission lines or the large antennas needed for the very low frequency transmissions used to communicate with submarines, are inherently vulnerable, although one can add circuitry to isolate them from other components in the system. Further, in the current U.S. C^3 system there are many components that are not hardened; it would be prohibitively expensive to harden the entire system. There is clearly no way of testing the entire system short of an actual battle to find out how much of it would survive.

The status of the C^3 systems of the Soviet Union in regard to EMP hardening is debated among defense strategists [23]. However, EMP is at some level a threat to the Soviet systems as well, with its vast array of computers, communications lines, antennas, power stations, and so forth.

In addition to EMP, there are other effects of radiation on integrated circuits, including total dose effects that can cause some integrated circuits to break down on exposure to 500 rads, and transient effects that can wipe out computer memories or cause semiconductor components to go into an abnormal conducting state [76]. Again, there are techniques to mitigate these effects, for example, by using technologies that are inherently harder, such

as integrated circuits based on gallium arsenide rather than silicon. However, as with EMP effects, there is clearly no way of realistically testing the entire system short of actual warfare.

A number of authorities have stated that the current U.S. C^3 system is deficient in regard to survivability (see, e.g., [17, 30, 48, 142]); some authors have gone further and stated that the current system is so vulnerable that a Soviet strike on communications facilities could wipe out the ability for the NCA to issue an order to retaliate.[10]

Limited nuclear war strategies put a greater burden on C^3 systems than do delayed retaliation strategies. Whether or not *any* C^3 system could support fighting a limited and centrally controlled nuclear war is a question hotly debated by experts in the field. Some experts assert that, although difficult, this is a goal worth pursuing: To maintain an effective deterrent, we must plan for nuclear conflict at any level of violence, and we must plan for ways to control such a conflict if it breaks out at a level below an all-out attack. For example, Charles Zraket, an executive vice-president of MITRE Corporation, writes, "The United States can achieve reliable deterrence only if it can ensure that it can retaliate discriminately and end a nuclear war as quickly as possible. Without this, deterrence is at the mercy of provocative rhetoric, threats of mutually assured destruction (MAD), or suicidal attacks" [142, p. 1306]. On the other hand, other experts, such as Desmond Ball, believe that the construction of a C^3 system to support fighting a limited nuclear war is not a reasonable option and should not be pursued [9, p. 38]. Supposing these latter experts are correct, then if a superpower attempts to fight a limited nuclear war, a likely outcome would be disconnected forces—each relying on its own damage assessments in deciding what to do next [21, pp. 98–128]. Under such conditions, it could be virtually impossible for an NCA to limit or stop the war, since even one of the isolated forces could continue it by launching another attack.

As discussed previously, means of communicating with the enemy would be important in trying to limit the scope of a nuclear war. Currently, the Hot Line linking Washington and Moscow terminates in the Pentagon and the Kremlin, neither of which is hardened against nuclear attack [80]. Various proposals have been made to add more redundancy to the Hot Line, perhaps with direct connections to the U.S. National Emergency Airborne Command Post[11] and its Soviet counterpart. Of course, nuclear war would have at least as severe an effect on U.S.–Soviet communications as on communications internal to the forces of one country.

An additional problem with limited nuclear war strategies is that, from a military point of view, one of the most efficient kinds of attack in a nuclear war is decapitation: attacks on political and military leadership and on command and control systems [119]. The "Fiscal Year 1984–1988 Defense Guidance" document cited previously states that U.S. nuclear war strategy is to be based on decapitation [61]. However, controlled nuclear war presumes that there is a NCA with which one can communicate. In the words of retired Air Force Lt. Gen. Brent Scowcroft [108, p. 95],

> there's a real dilemma here that we haven't sorted out. The kinds of controlled nuclear options to which we're moving presume communication with the Soviet Union; and yet, from a military point of view, one of the most efficient kinds of attack is against leadership and command and control systems. It's much easier than trying to take out each and every bit of the enemy's offensive forces. This is a dilemma that, I think, we still have not completely come to grips with.

It appears that Soviet military doctrine also calls for attacks on command and control systems at the outset of any strategic nuclear exchange to disrupt the enemy's forces and political and administrative control [9, p. 32].

It is beyond the scope of this article to evaluate closely the technical arguments regarding the hardening of C^3 systems, and in any event much of the information is classified. What *is* within the scope of this article is to observe that these C^3 systems are enormously complex and inherently untestable. In light of the previous discussions, there is thus considerable room for doubt that they would operate as planned in the event of a war.

FUTURE COMPUTER-CONTROLLED MILITARY SYSTEMS

If the arms race continues unabated, we will see increasing use of computer-controlled weapons systems that include little or no possibility of human intervention. Two specific projects are discussed here: the Strategic Defense Initiative (SDI), and the Strategic Computing Initiative (SCI).

[10] It does not necessarily follow, however, that there would be no retaliation; in a crisis forces might be put in a "fail-deadly" mode: If there was no communication for some period of time, retaliatory missiles would be launched. Such plans, if they exist, are highly classified. Some discussion of what such plans might be like may be found in Bracken [21]. On the same topic, former Secretary of Defense Harold Brown writes, "But a submarine-based missile could wipe out Washington with no more than ten minutes' warning, perhaps less. It is inappropriate to go into the details of the arrangements that have been made for such contingencies and thus suggest to the Soviets how to get around those arrangements. But one criterion for such arrangements ought properly to be that a decapitating attack should have the effect of making the response an all-out, unrestrained one" [27, p. 79].

[11] According to plan, this airborne command post would be in charge of the nation's nuclear forces if the land-based command posts were destroyed.

The Strategic Defense Initiative

In his now-famous speech of March 23, 1983, President Reagan presented a vision of the future: a technological means to escape the trap of MAD by the construction of a ballistic missile defense system that would render nuclear weapons "impotent and obsolete." That vision is now being pursued under the Strategic Defense Initiative Organization.

The SDI envisions a multilayer defense against nuclear ballistic missiles. The computer software to run such a defense would be the most complex ever built: A report by the Defensive Technologies Study Team, commissioned by the DoD to study the feasibility of such a system, estimates a system with 6–10 million lines of code [130, p. 45]. Enemy missiles would first be attacked in their boost phase, requiring action within 90 seconds or so of a detected launch. This time interval is so short that the human role in the system could be minimal at best, with virtually no possibility of decision making by national leaders. Although pieces of the system could be tested and simulation tests performed, it would be impossible to test the entire system under actual battle conditions short of fighting a nuclear war. It has been the universal experience in large computer systems that there is no substitute for testing under actual conditions of use. The SDI is the most extreme example so far of an untestable system. How could there be confidence that it would perform as intended? There are many other complicating factors, such as the constant need to update the software in response to new Soviet threats, and the difficulties of making updates to an operational space-based system. If the system is not trustworthy, it would seem unwise for the United States to abandon deterrence and nuclear missiles. How then could the SDI meet the goal of rendering nuclear weapons "impotent and obsolete?"

A more recent report by the Eastport Study Group [45], also commissioned by the DoD, stated that software considerations were the paramount problem in the SDI, and recommended a decentralized system rather than one requiring tight coordination. Nevertheless, the problems listed above hold whether a centralized or a decentralized architecture is used [98]. First, a battle station that is loosely coupled to the rest of the system must perform the functions of the whole system. The original arguments regarding complexity and untestability still apply. Second, individual battle stations would continue to interact. Some communication between stations would be needed for accurate tracking and for discrimination of warheads from decoys in the presence of noise. There would also be interactions through the weapons of one station and the sensors of another, for example, through the effect of noise generated by the destruction of a warhead. Again, we can have confidence that such interactions are well understood only by testing under realistic battle conditions.

If the SDI is deployed, there are two significant failure modes: failure of the system to stop a Soviet attack, and activation of the system due to a false alarm. Failure under attack would of course mean that a Soviet attack would *not* be stopped; if some plan is executed in which it is assumed that Soviet missiles have been rendered useless, the consequences would be devastating. Activation due to a false alarm would in itself not be as serious as inadvertently firing a missile. (However, some of the defensive systems being examined include nuclear weapons themselves, such as nuclear-pumped X-ray lasers. Further, it could be hard to distinguish quickly some defensive systems from offensive ones, such as those involving pop-up systems deployed on submarine-launched missiles.) The real danger is that the SDI would be integrated with the national strategic forces as a whole. An accidental SDI activation could trigger other responses, perhaps leading to a series of coupled escalating alerts on both sides or to other offensive actions. To lessen the chances of accidental activation in peacetime, Lieutenant General Abrahamson, the director of the Strategic Defense Initiative Organization, has suggested in testimony to Congress [133, pp. 704–705] that important parts of the system be placed under automatic control only during a crisis—but this is the worst possible time for an accidental activation to occur. There is a great deal more to be said on this topic, and the reader is referred to [79], [86], [97], and [98]. For two general overviews of SDI systems, see [1] and [4].

The Strategic Computing Initiative

In 1983 the Defense Advanced Research Projects Agency (DARPA) of the DoD proposed the SCI, a five-year, $600,000,000 effort to develop new computer-based military systems, emphasizing research in microelectronics and artificial intelligence [38]. Specific military applications to be built under the program are an autonomous robot vehicle capable of far-ranging reconnaissance and attack missions, an automated pilot's associate to aid fighter aircraft pilots, and a battle management system that can monitor incoming information, generate potential courses of action, disseminate orders, and compare actual events with those anticipated.

The integration of these weapons with nuclear war-fighting capabilities is planned: For example, the SCI states, "For certain space, air, and sea vehicles, the constraints and requirements will be even

higher and will include the capability to operate in high-radiation environments" [38, p. 23].

Whether or not artificial intelligence techniques are used (e.g., rule-based expert systems [65, 117] as is proposed in the SCI), the basic limitations discussed previously still hold. A battle-management system, for example, would in all probability give useful responses only in those situations for which rules were available—in other words, situations that had been anticipated by the experts who developed the rules. Again, as with any other computer system, we cannot be confident of its reliability until it has been extensively tested under conditions of actual use.

A more detailed analysis of the SCI may be found in an assessment by Computer Professionals for Social Responsibility [95]; a reply from the director of DARPA is in [35]. Another, more favorable assessment of the SCI was recently published by Mark Stefik [116].

CONCLUSIONS

How much reliance is it safe to place on life-critical computer systems, in particular, on nuclear weapons command and control systems? At present, a nuclear war caused by an isolated computer or operator error is probably not a primary risk, at least in comparison with other dangers. The most significant risk of nuclear war at present seems to come from the possibility of a *combination* of such events as international crises, mutually reinforcing alerts, computer system misdesign, computer failure, or human error.

A continuing trend in the arms race has been the deployment of missiles with greater and greater accuracies. This trend is creating increasing pressure to consider a launch-on-warning strategy. Such a strategy would, however, leave very little time to evaluate the warning and determine whether it were real or due to a computer or human error—we would be forced to put still greater reliance on the correct operation of the warning and command systems of the United States and the USSR. Deployment of very accurate missiles close to enemy territory exacerbates the problem.

C^3 systems should be such that leaders in both the United States and the USSR will not be forced into a "use it or lose it" situation, in which they feel they must launch a strike quickly lest their ability to retaliate is destroyed. Current war plans are more elaborate and include an array of options for flexible, limited nuclear responses. However, if a nuclear war should start, it is not at all clear that it would unfold according to these plans. We should always bear in mind that untested systems in a strange and hostile environment are not likely to perform reliably and as expected. In particular, it is impossible to determine exactly which components of a strategic command and control system would still work correctly after hostilities have commenced. This rules out strategies that depend on finely graded or complexly coordinated activities after the initial attack.

The construction of a ballistic missile defense system has been proposed. However, there could be no confidence that it would work as expected; in addition, its accidental activation during a crisis might trigger other hostilities. In the longer term, weapons systems equipped with extremely fast computers and using artificial intelligence techniques may result in battles (including nuclear ones) that must be largely controlled by computer.

Where then does that leave us? There is clearly room for technical improvements in nuclear weapons computer systems. I have argued, however, that adding more and more such improvements cannot ensure that they will always function correctly. The fundamental problems are due to untestability, limits of human decision making during high tension and crisis, and our inability to think through all the things that might happen in a complex and unfamiliar situation. We must recognize the limits of technology. The threat of nuclear war is a political problem, and it is in the political, human realm that solutions must be sought.

Acknowledgments. Many people have helped me in gathering information and in developing the ideas described here. Some of the original research was done in connection with a graduate seminar on Computer Reliability and Nuclear War held in the University of Washington Computer Science Department in Autumn 1982, and I thank the other participants in the seminar. Subsequently, a number of members of Computer Professionals for Social Responsibility have been generous with their help; I would particularly like to thank Guy Almes, Andrew Black, Gary Chapman, Calvin Gotlieb, Laura Gould, William Havens, Robert Henry, Jonathan Jacky, Cliff Johnson, Ira Kalet, Ed Lazowska, Peter Neumann, Severo Ornstein, David Parnas, Scott Rose, and Philip Wadler. Thanks also to Milton Leitenberg and Herbert Lin for expert advice on arms and arms control, and to Rob Kling for useful recommendations and suggestions in his role as area editor for this article.

REFERENCES

1. Adams, J.A., and Fischetti, M.A. *STAR WARS*—SDI: The grand experiment. *IEEE Spectrum 22*, 9 (Sept. 1985), 34–64.
2. Adrion, W.R., Branstad, M.A., and Cherniavsky, J.C. Validation, verification, and testing of computer software. *ACM Comput. Surv. 14*, 2 (June 1982), 159–192.
3. Albright, J. False missile alert required 3 minutes to cancel. *San Jose Mercury* (June 15, 1980), 1 H.

4. American Academy of Arts and Sciences. Weapons in space. Vol. I: Concepts and technologies. *Daedalus 114*, 2 (Spring 1985), 9–189.
5. Anderson, T., and Randell, B., Eds. *Computing Systems Reliability: An Advanced Course.* Cambridge University Press, Cambridge, Mass., 1979.
6. Angus, J.E. The application of software reliability models to a major C³I system. In *Proceedings of the Annual Reliability and Maintainability Symposium* (San Francisco, Calif., Jan. 24–26). IEEE Press, New York, 1984, pp. 268–274.
7. Arkin, W.M. Nuclear weapon command, control, and communications. In *World Armaments and Disarmament: SIPRI Yearbook 1984*, F. Blackaby, Ed. Taylor and Francis, London, 1984, pp. 455–516.
8. Association for Computing Machinery. Proceedings of VERkshop III—A formal verification workshop. *Softw. Eng. Notes 10*, 4 (1985).
9. Ball, D. Can nuclear war be controlled? Adelphi Pap. 169. International Institute for Strategic Studies, London, 1981.
10. Ball, D. The Soviet strategic C³I system. In *C³I Handbook.* EW Communications, Palo Alto, Calif., 1986, pp. 206–216.
11. Balzer, R., Cheatham, T.E., and Green, C. Software technology in the 1990's: Using a new paradigm. *Computer 16*, 11 (Nov. 1983), 39–45.
12. Barstow, D.R., Shrobe, H.E., and Sandewall, E., Eds. *Interactive Programming Environments.* McGraw-Hill, New York, 1984.
13. Barstow, D.R. Domain-specific automatic programming. *IEEE Trans. Softw. Eng. SE-11*, 11 (Nov. 1985), 1321–1336.
14. Benzel, T.C.V., and Tavilla, D.A. Trusted software verification: A case study. In *Proceedings of the 1985 Symposium on Security and Privacy* (Oakland, Calif., Apr. 22–24). IEEE Press, New York, 1985, pp. 14–31.
15. Bereanu, B. Self-activation of the world nuclear weapons system. *J. Peace Res. 20*, 1 (1983), 49–57.
16. Berkeley, E.C. *The Computer Revolution.* Doubleday, New York, 1962.
17. Blair, B.G. *Strategic Command and Control.* Brookings Institution, Washington, D.C., 1985.
18. Bloomfield, L.P. Nuclear crisis and human frailty. *Bull. At. Sci. 41*, 9 (Oct. 1985), 26–30.
19. Boehm, B.W. *Software Engineering Economics.* Prentice-Hall, Englewood Cliffs, N.J., 1981.
20. Boyer, R.S., and Moore, J.S., Eds. *The Correctness Problem in Computer Science.* Academic Press, New York, 1981.
21. Bracken, P. *The Command and Control of Nuclear Forces.* Yale University Press, New Haven, Conn., 1983.
22. Broad, W.J. Computers and the U.S. military don't mix. *Science 207*, 4436 (Mar. 14, 1980), 1183–1187.
23. Broad, W.J. Nuclear pulse (I): Awakening to the chaos factor. *Science 212*, 4498 (May 29, 1981), 1009–1012.
24. Broad, W.J. Nuclear pulse (II): Ensuring delivery of the doomsday signal. *Science 212*, 4499 (June 5, 1981), 1116–1120.
25. Broad, W.J. Nuclear pulse (III): Playing a wild card. *Science 212*, 4500 (June 12, 1981), 1248–1251.
26. Brooks, J. NORAD computer systems are dangerously obsolete. House Rep. 97-449, Committee on Government Operations, United States House of Representatives, Washington, D.C., Mar. 8, 1982.
27. Brown, H. *Thinking about National Security: Defense and Foreign Policy in a Dangerous World.* Westview Press, Boulder, Colo., 1983.
28. Bunn, M., and Tsipis, K. The uncertainties of a preemptive nuclear attack. *Sci. Am. 249*, 5 (Nov. 1983), 38–47.
29. Burt, R. False nuclear alarms spur urgent effort to find faults. *New York Times* (June 13, 1980), A16.
30. Carter, A.B. The command and control of nuclear war. *Sci. Am. 252*, 1 (Jan. 1985), 32–39.
31. Carter, W.C. Hardware fault tolerance. In *Computing Systems Reliability: An Advanced Course*, T. Anderson and B. Randell, Eds. Cambridge University Press, Cambridge, Mass., 1979, Chap. 6, pp. 211–263.
32. Cockburn, A. *The Threat: Inside the Soviet Military Machine.* Random House, New York, 1983.
33. Comptroller General of the United States. The world wide military command and control system—Major changes needed in its automated data processing management and direction. Rep. LCD-80-22, Comptroller General of the United States, Washington, D.C., Dec. 14, 1979.
34. Comptroller General of the United States. *NORAD's Missile Warning System: What Went Wrong?* United States General Accounting Office, Washington, D.C., May 15, 1981.
35. Cooper, R.S. Letter to the editor. *Bull. At. Sci. 41*, 1 (Jan. 1985), 54–55.
36. Currit, P.A., Dyer, M., and Mills, H.D. Certifying the reliability of software. *IEEE Trans. Softw. Eng. SE-12*, 1 (Jan. 1986), 3–11.
37. Davis, N.C., and Goodman, S.E. The Soviet bloc's unified system of computers. *ACM Comput. Surv. 10*, 2 (June 1978), 93–122.
38. Defense Advanced Research Projects Agency. *Strategic Computing—New-Generation Computing Technology: A Strategic Plan for its Development and Application to Critical Problems in Defense.* U.S. Department of Defense, Arlington, Va., Oct. 28, 1983.
39. Defense Nuclear Agency. *EMP Awareness Handbook.* Defense Nuclear Agency, Washington, D.C., 1971.
40. Deutsch, M.S. *Software Verification and Validation: Realistic Project Approaches.* Prentice-Hall, Englewood Cliffs, N.J., 1982
41. Doder, D. Soviets said to consider faster nuclear missile launch in crisis. *Washington Post* (Apr. 11, 1982), A5.
42. Doder, D. Kremlin defense official warns of policy shift to quicken nuclear response. *Washington Post* (July 13, 1982), A1a.
43. Druffel, L.E., Redwine, S.T., and Riddle, W.E. The STARS program: Overview and rationale. *Computer 16*, 11 (Nov. 1983), 21–29.
44. Dumas, L.J. Human fallibility and weapons. *Bull. At. Sci. 36*, 9 (Nov. 1980), 15–20.
45. Eastport Study Group. *Summer Study 1985: A Report to the Director, Strategic Defense Initiative Organization.* Strategic Defense Initiative Organization, Dec. 1985.
46. Emelyanov, V.S. The possibility of an accidental nuclear war. In *The Arms Race at a Time of Decision*, J. Rotblat and A. Pascolini, Eds. Macmillan, New York, 1984, Chap. 9, pp. 73–79.
47. Fairley, R.E. *Software Engineering Concepts.* McGraw-Hill, New York, 1985.
48. Ford, D. *The Button.* Simon and Schuster, New York, 1985.
49. Fox, R. *Software and Its Development.* Prentice-Hall, Englewood Cliffs, N.J., 1982.
50. Frei, D. *Risks of Unintentional Nuclear War.* Allanheld, Osmun and Co., Totowa, N.J., 1983.
51. Garman, J.R. The "bug" heard 'round the world. *Softw. Eng. Notes 6*, 5 (Oct. 1981), 3–10.
52. Garwin, R. Launch under attack to redress minuteman vulnerability? *Int. Secur. 4*, 3 (Winter 1979), 117–139.
53. Gelb, L.H. Soviet marshal warns the U.S. on its missiles. *New York Times* (Mar. 17, 1983), A1.
54. George, A., and Smoke, R. *Deterrence in American Foreign Policy.* Columbia University Press, New York, 1974.
55. Glasstone, S., and Dolan, P.J. *The Effects of Nuclear Weapons.* U.S. Department of Defense, Washington, D.C., 1977.
56. Goodman, S.E. Computing and the development of the Soviet economy. A Compendium of Papers Submitted to the Joint Economic Committee of the Congress of the United States, Washington, D.C., Oct. 10, 1979.
57. Gray, C.S., and Payne, K. Victory is possible. *Foreign Policy 39* (Summer 1980), 14–27.
58. Gumble, B. Air Force upgrading defenses at NORAD. *Def. Electron. 17*, 8 (Aug. 1985), 86–108.
59. Halloran, R. U.S. aides recount moments of false missile alert. *New York Times* (Dec. 16, 1979), 25.
60. Halloran, R. Computer error falsely indicates a Soviet attack. *New York Times* (June 6, 1980), 14.
61. Halloran, R. Pentagon draws up first strategy for fighting a long nuclear war. *New York Times* (May 30, 1982), 1:1.
62. Halloran, R. Weinberger confirms new strategy on atom war. *New York Times* (June 4, 1982), A10.
63. Halloran, R. Shift of strategy on missile attack hinted by Weinberger and Vessey. *New York Times* (May 6, 1983), 1:1.
64. Hart, G., and Goldwater, B. *Recent False Alerts from the Nation's Missile Attack Warning System.* United States Senate, Committee on Armed Services, Washington, D.C., 1980.
65. Hayes-Roth, F., Waterman, D., and Lenat, D. *Building Expert Systems.* Addison-Wesley, Reading, Mass., 1983.
66. Henderson, P., Ed. *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.* ACM, New York, 1984.
67. Howard, M. On fighting a nuclear war. *Int. Secur. 5*, 4 (Spring 1981), 3–17.
68. Howden, W.E. Validation of scientific programs. *ACM Comput. Surv. 14*, 2 (June 1982), 193–227.
69. Hubbell, J.C. You are under attack! The strange incident of October 5. *Reader's Dig. 78*, 468 (Apr. 1961), 37–41.
70. International Institute for Strategic Studies. *The Military Balance 1983–1984.* International Institute for Strategic Studies, London, 1983.
71. Jensen, R.W., and Tonies, C.C. *Software Engineering.* Prentice-Hall, Englewood Cliffs, N.J., 1979.
72. Kahn, H. *On Thermonuclear War.* Princeton University Press, Princeton, N.J., 1960.
73. Kaplan, F. *The Wizards of Armageddon.* Simon and Schuster, New York, 1983.
74. Kissinger, H.A. *Nuclear Weapons and Foreign Policy.* Harper, New York, 1957.

75. Kling, R. Defining the boundaries of computing across complex organizations. In *Critical Issues in Information Systems Research*, R. Boland and R. Hirschheim, Eds. Wiley, New York, 1987.
76. Lerner, E.J. Electronics and the nuclear battlefield. *IEEE Spectrum 19*, 10 (Oct. 1982), 64–65.
77. Leveson, N.G. Software safety: Why, what, and how. Rep. 86-04, Dept. of Information and Computer Science, Univ. of California, Irvine, Feb. 1986.
78. Lieblein, E. The Department of Defense Software Initiative—A status report. *Commun. ACM 29*, 8 (Aug. 1986), 734–744.
79. Lin, H. Software for ballistic missile defense. Rep. C/85-2, Center for International Studies, MIT, Cambridge, Mass., June 1985.
80. Martin, R. Stopping the unthinkable: C^3I dimensions of terminating a "limited" nuclear war. Rep. P-82-3, Center for Information Policy Research, Harvard Univ., Cambridge, Mass., Apr. 1982.
81. Meyers, G.J. *Software Reliability: Principles and Practices.* Wiley, New York, 1976.
82. Meyers, G.J. *The Art of Software Testing.* Wiley, New York, 1979.
83. Miller, G.E. Existing systems of command and control. In *The Dangers of Nuclear War*, Griffiths, Franklyn, and Polanyi, Eds. University of Toronto Press, Toronto, Ontario, 1979, pp. 50–66.
84. MITRE Corp. National security issues symposium: Strategic nuclear policies, weapons, and the C^3 connection. Doc. M82-30, MITRE Corp., Bedford, Mass., 1982.
85. Morgan, P.M. *Deterrence: A Conceptual Analysis.* 2nd ed. Sage, Beverley Hills, Calif., 1983.
86. Nelson, G., and Redell, D. The Star Wars computer systems. *Abacus 3*, 2 (Winter 1986), 8–20.
87. Neumann, P.G. An editorial on software correctness and the social process. *Softw. Eng. Notes 4*, 2 (Apr. 1979), 3–4.
88. Neumann, P.G. Letter from the editor. *Softw. Eng. Notes 10*, 1 (Jan. 1985), 3–11.
89. *New York Times.* Moon stirs scare of missile attack. *New York Times* (Dec. 8, 1960), 71:2.
90. *New York Times.* False alarm on attack sends fighters into sky. *New York Times* (Nov. 10, 1979), 21.
91. *New York Times.* Missile alerts traced to 46 cent item. *New York Times* (June 18, 1980), 16.
92. *New York Times.* Nevada governor says errors led to flooding. *New York Times* (July 4, 1983), I-10.
93. Nye, J.S., Allison, G.T., and Carnesale, A. Analytic conclusions: Hawks, doves, and owls. In *Hawks, Doves, and Owls*, G.T. Allison, A. Carnesale, and J.S. Nye, Eds. Norton, New York, 1985. Chap. 8, pp. 206–222.
94. Office of Technology Assessment. MX missile basing—Launch under attack. Office of Technology Assessment, Washington, D.C., 1981.
95. Ornstein, S., Smith, B.C., and Suchman, L. Strategic computing: An assessment. *Bull. At. Sci. 40*, 10 (Dec. 1984), 11–15.
96. Parnas, D.L., Clements, P.C., and Weiss, D.M. The modular structure of complex systems. In *Proceedings of the 7th International Conference on Software Engineering* (Orlando, Fla., Mar. 26–29). IEEE Press, New York, 1984, pp. 408–417.
97. Parnas, D.L. Software aspects of strategic defense systems. *Am. Sci. 73*, 5 (Sept.–Oct. 1985), 432–440.
98. Parnas, D.L. SDI: A violation of professional responsibility. *Abacus 4*, 2 (Winter 1987), 46–52.
99. Partsch, H., and Steinbruggen, R. Program transformation systems. *ACM Comput. Surv. 15*, 3 (Sept. 1983), 199–236.
100. Perrow, C. Normal accident at Three Mile Island. *Society 18*, 5 (July–Aug. 1981), 17–26.
101. Perrow, C. *Normal Accidents: Living with High-Risk Technologies.* Basic Books, New York, 1984.
102. Pollack, A. Trust in computers raising risk of errors and sabotage. *New York Times* (Aug. 22, 1983), 1.
103. Pringle, P., and Arkin, W. *S.I.O.P.: The Secret U.S. Plan for Nuclear War.* Norton, New York, 1983.
104. Ricketts, L.W. *Fundamentals of Nuclear Hardening of Electronic Equipment.* Wiley, New York, 1972.
105. Ricketts, L.W., Bridges, J.E., and Miletta, J. *EMP Radiation and Protective Techniques.* Wiley, New York, 1976.
106. Rogovin, M., and Frampton, G.T. *Three Mile Island: A Report to the Commissioners and to the Public.* Nuclear Regulatory Commission Special Inquiry Group, U.S. Nuclear Regulatory Commission, Washington, D.C., 1980.
107. Rosen, E. Vulnerabilities of network control protocols: An example. *Softw. Eng. Notes 6*, 1 (Jan. 1981), 6–8.
108. Scowcroft, B. C^3 systems for the president and military commanders. In *National Security Issues Symposium: Strategic Nuclear Policies, Weapons, and the C^3 Connection*, D.M. Ace, Ed. MITRE Corp., Bedford, Mass., 1982, pp. 93–97.
109. Scowcroft, B. *Report of the President's Commission on Strategic Forces.* U.S. Department of Defense, Washington, D.C., Apr. 1983.
110. *Seattle Post-Intelligencer.* Russia puts more N-arms off U.S. Coast. *Seattle Post-Intelligencer* (May 21, 1984), A2.
111. Siewiorek, D.P., and Swarz, R.S. *The Theory and Practice of Reliable System Design.* Digital Press, Bedford, Mass., 1982.
112. Silverman, J.M. Reflections on the verification of the security of an operating system. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles* (Oct.). ACM, New York, 1983, pp. 143–154.
113. Smith, B.C. The limits of correctness. *ACM SIGCAS Newsl. 14*, 4 and *15*, 1–3 (Winter–Fall 1985), 18–26.
114. Sokolovskiy, V.D. *Soviet Military Strategy.* Edited, with an analysis and commentary, by H. Fast Scott. Crane, Russak and Co., New York, 1975.
115. Sommerville, I. *Software Engineering.* Addison-Wesley, Reading, Mass., 1982.
116. Stefik, M. Strategic computing at DARPA: Overview and assessment. *Commun. ACM 28*, 7 (July 1985), 690–704.
117. Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F., and Sacerdoti, E. The organization of expert systems, a tutorial. *Artif. Intell. 18*, 2 (Mar. 1982), 135–173.
118. Stein, D.L. Electromagnetic pulse—The uncertain certainty. *Bull. At. Sci. 39*, 3 (Mar. 1983), 52–56.
119. Steinbruner, J.D. Nuclear decapitation. *Foreign Policy 45* (Winter 1981–1982), 16–28.
120. Steinbruner, J.D. Launch under attack. *Sci. Am. 250*, 1 (Jan. 1984), 37–47.
121. Stevens, R.T. Testing the NORAD command and control system. *IEEE Trans. Syst. Sci. Cybern. SSC-4*, 1 (Mar. 1968), 47–51.
122. Stockholm International Peace Research Institute. *World Armaments and Disarmament: SIPRI Yearbook 1983.* Taylor and Francis, London, 1983.
123. Stockholm International Peace Research Institute. *World Armaments and Disarmament: SIPRI Yearbook 1984.* Taylor and Francis, London, 1984.
124. Tasky, K. Soviet technology gap and dependence on the west: The case of computers. A Compendium of Papers Submitted to the Joint Economic Committee of the Congress of the United States, Washington, D.C., Oct. 10, 1979.
125. U.S. Department of Defense. *Military Standardization Handbook: Reliability Prediction of Electronic Equipment. MIL-STD-HDBK-217C.* Notice 1 ed. U.S. Department of Defense, Washington, D.C., 1980.
126. U.S. Department of Defense. *Modernization of the WWMCCS Information System (WIS).* The Assistant Secretary of Defense (CCCI) with the assistance of the WWMCCS System Engineer (DCA). U.S. Department of Defense, Washington, D.C., July 31, 1982.
127. U.S. Department of Defense. *Soviet Military Power.* 2nd ed. U.S. Government Printing Office, Washington, D.C., 1983.
128. U.S. Department of Defense. Software technology for adaptable, reliable systems. *Softw. Eng. Notes 8*, 2 (Apr. 1983), 55–84.
129. U.S. Department of Defense. *Soviet Military Power.* 3rd ed. U.S. Government Printing Office, Washington, D.C., 1984.
130. U.S. Department of Defense. *Report of the Study on Eliminating the Threat Posed by Nuclear Ballistic Missiles.* Vol. 5, *Battle Management, Communications, and Data Processing.* U.S. Department of Defense, Washington, D.C., Feb. 1984.
131. U.S. Department of Defense. *DOD-STD-2167: Military Standard Defense System Software Development.* U.S. Department of Defense, Washington, D.C., 1985.
132. U.S. House of Representatives. *Failures of the North American Aerospace Defense Command's (NORAD) Attack Warning System.* Hearings before a Subcommittee of the Committee on Government Operations, U.S. House of Representatives, 97th Congress, 1st session, May 19 and 20, 1981.
133. U.S. House of Representatives. *Hearings, Department of Defense Appropriations for 1985.* Committee on Appropriations, Subcommittee on the Department of Defense, U.S. Government Printing Office, Washington, D.C., 1984.
134. U.S. House of Representatives. *Our Nation's Nuclear Warning System: Will It Work If We Need It?* Hearings before a Subcommittee of the Committee on Government Operations, U.S. House of Representatives, 99th Congress, 1st session, Sept. 26, 1985.
135. U.S. Senate Committee on Armed Services. *Hearings, Department of Defense Authorization for Appropriations for FY 1982, Part 7.* U.S. Government Printing Office, Washington, D.C., 1981.
136. U.S. Senate Committee on Armed Services. *Hearings, Department of Defense Authorization for Appropriations for FY 1984, Part 5.* U.S. Government Printing Office, Washington, D.C., 1983.
137. Van Evera, S. The cult of the offensive and the origins of the First World War. *Int. Secur. 9*, 1 (Summer 1984), 58–107.

138. Waltz, E.L. Data fusion for C³I systems. In *C³I Handbook*. EW Communications, Palo Alto, Calif., 1986, pp. 217–226.
139. *Washington Post*. Computer again gives signal of false Soviet attack. *Washington Post* (June 8, 1980), A7.
140. *Washington Post*. Soviet warns of automatic retaliation against new U.S. missiles. *Washington Post* (May 18, 1983), A12.
141. Weinberger, C.W. *Report of the Secretary of Defense to the Congress on the FY 1985 Budget, FY 1986 Authorization Request and FY 1985–89 Defense Programs*. U.S. Government Printing Office, Washington, D.C., 1984.
142. Zraket, C.A. Strategic command, control, communications, and intelligence. *Science 224*, 4655 (June 22, 1984), 1306–1311.

CR Categories and Subject Descriptors: J.7 [**Computers in Other Systems**]: *command and control; military*; K.4.2 [**Computers and Society**]: Social Issues

General Terms: Reliability

Additional Key Words and Phrases: Battle management systems, false alerts, launch-on-warning, life-critical computer systems, limited nuclear war, Strategic Computing Initiative, Strategic Defense Initiative, unintentional nuclear war

Author's Present Address: Alan Borning, Dept. of Computer Science, FR-35, University of Washington, Seattle, WA 98195.