Which argument is stronger? Let's focus on one of the oldest NCIC databases: the database of stolen vehicles. The total amount of harm caused to society by automobile theft is great. Over one million automobiles are stolen in the United States every year. Victims of car theft are subjected to emotional stress, may sustain a financial loss, and can spend a lot of time trying to recover or replace the vehicle. In addition, the prevalence of automobile theft harms everyone who owns a car by raising insurance rates. In the past, car thieves could reduce the probability that a stolen car would be recovered by transporting it across a state line, but the NCIC database contains information about stolen vehicles throughout the United States, and it has enabled law enforcement officials to identify cars stolen anywhere in the nation. At the present time, just over half of all stolen vehicles are recovered. If we make the conservative estimate that the NCIC has increased the percentage of recovered cars by just 10 percent, more than 50,000 additional cars are being returned to their owners each year, a significant benefit. On the other hand, if an error in the NCIC stolen car database leads to a false arrest, the harm caused to the innocent driver is great. However, there are only a few stories of false arrests stemming from errors in the NCIC stolen car database. The total amount of benefit derived from the NCIC database of stolen automobiles appears to be much greater than the total amount of harm it has caused. We conclude the creation and maintenance of this database has been the right course of action.

8.3 Software and Billing Errors

Even if the data entered into a computer are correct, the system may still produce the wrong result or collapse entirely if there are errors in the computer programs manipulating the data. Newspapers are full of stories about software bugs or "glitches." Here is a selection of stories that have appeared in print.

8.3.1 Errors Leading to System Malfunctions

Linda Brooks of Minneapolis, Minnesota, opened her mail on July 21, 2001, and found a phone bill for $57,346.20. A bug in Qwest's billing software caused it to charge some customers as much as $600 per minute for the use of their cell phones. About 1.4 percent of Qwest's customers, 14,000 in all, received incorrect bills. A Qwest spokesperson said the bug was in a newly installed billing system [6].

The U.S. Department of Agriculture implemented new livestock price-reporting guidelines after discovering that software errors had caused the USDA to understate the prices meat packers were receiving for beef. Since beef producers and packers negotiate cattle contracts based on the USDA price reports, the errors cost beef producers between $15 and $20 million [7].

In 1996, a software error at the U.S. Postal Service caused it to return to the senders two weeks' worth of mail addressed to the Patent and Trademark Office. In all, 50,000 pieces of mail were returned to sender [8].

A University of Pittsburgh study revealed that for most students, computer spelling and grammar error checkers actually increased the number of errors they made [9, 10].

Between September 2008 and May 2009, hundreds of families living in public housing in New York City were charged too much rent because of an error in the program that calculated their monthly bills. For nine months, the New York City Housing Authority did not take seriously the renters' complaints that they were being overcharged. Instead, it took to court many of the renters who did not make the higher payments and threatened them with eviction [11].

In 2010, about 450 California prison inmates with a "high risk of violence" were mistakenly released as part of a program meant to reduce prison overcrowding. California officials could not return any of them to prison or put them on supervised parole because they had already been granted "nonrevocable parole" [12].

8.3.2 Errors Leading to System Failures

On the first day a new, fully computerized ambulance dispatch system became operational in the City of London, people making emergency calls were put on hold for up to 30 minutes, the system lost track of some calls, and ambulances took up to three hours to respond. As many as 20 people died because ambulances did not arrive in time [13].

A software error led the Chicago Board of Trade to suspend trading for an hour on January 23, 1998. Another bug caused it to suspend trading for 45 minutes on April 1, 1998. In both cases, the temporary shutdown of the trading caused some investors to lose money [14]. System errors caused trading on the London International Financial Futures and Options Exchange to be halted twice within two weeks in May 1999. The second failure idled dealers for an hour and a half [16].

Thailand's finance minister was trapped inside his BMW limousine for 10 minutes when the on-board computer crashed, locking all doors and windows and turning off the air-conditioning. Security guards had to use sledge hammers to break a window, enabling Suchart Jaovisidha and his driver to escape [15].

Japan's air traffic control system went down for an hour on the morning of March 1, 2003, delaying departures for hours. The backup system failed at the same time as the main system, which was out of commission for four hours. Airports kept in touch via telephone, and no passengers were put at risk. However, some flights were delayed over two hours, and 32 domestic flights had to be canceled [17].

A new laboratory computer system at Los Angeles County+USC Medical Center became backlogged the day after it was turned on. For several hours on both April 16 and 17, 2003, emergency room doctors told the County of Los Angeles to stop sending ambulances, because the doctors could not get access to the laboratory results they needed. "It's almost like practicing Third World medicine," said Dr. Amanda Garner. "We rely so much on our computers and our fast-world technology that we were almost blinded" [18].

FIGURE 8.1 Comair cancelled all of its flights on Christmas Day, 2004, because the computer system that assigned crews to flights failed. (AP Photo/Al Behrman, File)

Comair, a subsidiary of Delta Air Lines, canceled all 1,100 of its flights on Christmas Day, 2004, because the computer system that assigns crews to flights stopped running (Figure 8.1). Airline officials said the software could not handle the large number of flight cancellations caused by bad weather on December 23 and 24. About 30,000 travelers in 118 cities were affected by the flight cancellations [19].

In August 2005 the passengers on a Malaysia Airlines flight from Perth, Australia, to Kuala Lumpur, Malaysia, suddenly found themselves on a roller coaster–like ride seven miles above the Indian Ocean. When the Boeing 777 unexpectedly began a rapid climb, the pilot disconnected the autopilot, but it took him 45 seconds to regain control of the jet. The plane zoomed upward, downward, and then upward a second time before leveling out. After an investigation, Boeing reported that a software error had caused the flight computers to receive faulty information about the plane's speed and acceleration. In addition, another error had caused the flight computers to fail to respond immediately to the pilot's commands [20].

8.3.3 Analysis: E-Retailer Posts Wrong Price, Refuses to Deliver

Amazon.com shut down its British Web site on March 13, 2003, after a software error led it to offer iPaq handheld computers for £7 instead of the correct price of about £275. Before Amazon.com shut down the site, electronic bargain hunters had flocked to Amazon.com's Web site, some of them ordering as many as ten iPaQs [21]. Amazon said

elections officials. "Open source advocates and paper trail champions want to steer e-voting off a cliff. Rather than demanding utopian machines and spreading conspiracy theories for political gain, they should re-focus their energy in a way that actually helps American voters" [45].

Nevertheless, some states are having second thoughts about DRE voting machines. In May 2007 Florida's legislature voted to replace DRE voting machines with optical scan ballots. Voters select candidates by filling in bubbles next to their names, and optical scanning machines count the marked ballots. This approach leaves a paper audit trail that makes possible manual recounts in disputed elections [46].

8.5 Therac-25

Soon after German physicist Wilhelm Roentgen discovered the x-ray in 1895, physicians began using radiation to treat cancer. Today, between 50 and 60 percent of cancer patients are treated with radiation, either to destroy cancer cells or relieve pain. Linear accelerators create high-energy electron beams to treat shallow tumors and x-ray beams to reach deeper tumors.

The Therac-25 linear accelerator was notoriously unreliable. It was not unusual for the system to malfunction 40 times a day. We devote an entire section to telling the story of the Therac-25 because it is a striking example of the harm that can be caused when the safety of a system relies solely upon the quality of its embedded software.

In a 20-month period between June 1985 and January 1987, the Therac-25 administered massive overdoses to six patients, causing the deaths of three of them. While 1987 may seem like the distant past to many of you, it does give us the advantage of 20/20 hindsight. The entire story has been thoroughly researched and documented [47]. Failures of computerized systems continue to this day, but they have not yet been fully played out and analyzed.

8.5.1 Genesis of the Therac-25

Atomic Energy of Canada Limited (AECL) and the French corporation CGR cooperated in the 1970s to build two linear accelerators: the Therac-6 and the Therac-20. Both the Therac-6 and the Therac-20 were modernizations of older CGR linear accelerators. The distinguishing feature of the Therac series was the use of a DEC PDP 11 minicomputer as a "front end." By adding the computer, the linear accelerators were easier to operate. The Therac-6 and the Therac-20 were actually capable of working independently of the PDP 11, and all of their safety features were built into the hardware.

After producing the Therac-20, AECL and CGR went their separate ways. AECL moved ahead with the development and deployment of a next-generation linear accelerator called the Therac-25. Like the Therac-6 and the Therac-20, the Therac-25 made use of a PDP 11. Unlike its predecessor machines, however, AECL designed the PDP 11 to be an integral part of the device; the linear accelerator was incapable of operating without the computer. This design decision enabled AECL to reduce costs by replacing

some of the hardware safety features of the Therac-20 with software safety features in the Therac-25.

AECL also decided to reuse some of the Therac-6 and Therac-20 software in the Therac-25. Code reuse saves time and money. Theoretically, "tried and true" software is more reliable than newly written code, but as we shall see, that assumption was invalid in this case.

AECL shipped its first Therac-25 in 1983. In all, it delivered 11 systems in Canada and the United States. The Therac-25 was a large machine that was placed in its own room. Shielding in the walls, ceiling, and floor of the room prevented outsiders from being exposed to radiation. A television camera, microphone, and speaker in the room allowed the technician in an adjoining room to view and communicate with the patient undergoing treatment.

8.5.2 Chronology of Accidents and AECL Responses

MARIETTA, GEORGIA, JUNE 1985

A 61-year-old breast cancer patient was being treated at the Kennestone Regional Oncology Center. After radiation was administered to the area of her collarbone, she complained that she had been burned.

The Kennestone physicist contacted AECL and asked if it was possible that the Therac-25 had failed to diffuse the electron beam. Engineers at AECL replied that this could not happen.

The patient suffered crippling injuries as a result of the overdose, which the physicist later estimated was 75 to 100 times too large. She sued AECL and the hospital in October 1985.

HAMILTON, ONTARIO, JULY 1985

A 40-year-old woman was being treated for cervical cancer at the Ontario Cancer Foundation. When the operator tried to administer the treatment, the machine shut down after five seconds with an error message. According to the display, the linear accelerator had not yet delivered any radiation to the patient. Following standard operating procedure, the operator typed "P" for "proceed." The system shut down in the same way, indicating that the patient had not yet received a dose of radiation. (Recall it was not unusual for the machine to malfunction several dozen times a day.) The operator typed "P" three more times, always with the same result, until the system entered "treatment suspend" mode.

The operator went into the room where the patient was. The patient complained that she had been burned. The lab called in a service technician, who could find nothing wrong with the machine. The clinic reported the malfunction to AECL.

When the patient returned for further treatment three days later, she was hospitalized for a radiation overdose. It was later estimated that she had received between 65 and 85 times the normal dose of radiation. The patient died of cancer in November 1985.

FIRST AECL INVESTIGATION, JULY–SEPTEMBER 1985

After the Ontario overdose, AECL sent out an engineer to investigate. While the engineer was unable to reproduce the overdose, he did uncover design problems related to a microswitch. AECL introduced hardware and software changes to fix the microswitch problem.

YAKIMA, WASHINGTON, DECEMBER 1985

The next documented overdose accident occurred at Yakima Valley Memorial Hospital. A woman receiving a series of radiation treatments developed a strange reddening on her hip after one of the treatments. The inflammation took the form of several parallel stripes. The hospital staff tried to determine the cause of the unusual stripes. They suspected the pattern could have been caused by the slots in the accelerator's blocking trays, but these trays had already been discarded by the time the staff began their investigation. After ruling out other possible causes for the reaction, the staff suspected a radiation overdose and contacted AECL by letter and by phone.

AECL replied in a letter that neither the Therac-25 nor operator error could have produced the described damage. Two pages of the letter explained why it was technically impossible for the Therac-25 to produce an overdose. The letter also claimed that no similar accidents had been reported.

The patient survived, although the overdose scarred her and left her with a mild disability.

TYLER, TEXAS, MARCH 1986

A male patient came to the East Texas Cancer Center (ETCC) for the ninth in a series of radiation treatments for a cancerous tumor on his back. The operator entered the treatment data into the computer. She noticed that she had typed "X" (for x-ray) instead of "E" (for electron beam). This was a common mistake, because x-ray treatments are much more common. Being an experienced operator, she quickly fixed her mistake by using the up arrow key to move the cursor back to the appropriate field, changing the "X" to an "E" and moving the cursor back to the bottom of the screen. When the system displayed "beam ready," she typed "B" (for beam on). After a few seconds, the Therac-25 shut down. The console screen contained the message "Malfunction 54" and indicated a "treatment pause," a low-priority problem. The dose monitor showed that the patient had received only 6 units of treatment rather than the desired 202 units. The operator hit the "P" (proceed) key to continue the treatment.

The cancer patient and the operator were in adjoining rooms. Normally a video camera and intercom would enable the operator to monitor her patients. However, at the time of the accident neither system was operational.

The patient had received eight prior treatments, so he knew something was wrong as soon as the ninth treatment began. He was instantly aware of the overdose—he felt as if someone had poured hot coffee on his back or given him an electric shock. As he tried to get up from the table, the accelerator delivered its second dose, which hit him in the arm. The operator became aware of the problem when the patient began pounding on the

door. He had received between 80 and 125 times the prescribed amount of radiation. He suffered acute pain and steadily lost bodily functions until he died from complications of the overdose five months later.

SECOND AECL INVESTIGATION, MARCH 1986

After the accident, the ETCC shut down its Therac-25 and notified AECL. AECL sent out two engineers to examine the system. Try as they might, they could not reproduce Malfunction 54. They told the physicians it was impossible for the Therac-25 to overdose a patient, and they suggested that the patient may have received an electrical shock due to a fault in the hospital's electrical system.

The ETCC checked out the electrical system and found no problems with it. After double-checking the linear accelerator's calibration, they put the Therac-25 back into service.

TYLER, TEXAS, APRIL 1986

The second Tyler, Texas, accident was virtually a replay of the prior accident at ETCC. The same technician was in control of the Therac-25, and she went through the same process of entering x-ray when she meant electron beam, then going back and correcting her mistake. Once again, the machine halted with a Malfunction 54 shortly after she activated the electron beam. This time, however, the intercom was working, and she rushed to the accelerator when she heard the patient moan. There was nothing she could do to help him. The patient had received a massive dose of radiation to his brain, and he died three weeks later.

After the accident, ETCC immediately shut down the Therac-25 and contacted AECL again.

YAKIMA, WASHINGTON, JANUARY 1987

A second patient was severely burned by the Therac-25 at Yakima Valley Memorial Hospital under circumstances almost identical to those of the December 1985 accident. Four days after the treatment, the patient's skin revealed a series of parallel red stripes—the same pattern that had perplexed the radiation staff in the case of the previous patient. This time, the staff members were able to match the burns to the slots in the Therac-25's blocking tray. The patient died three months later.

THERAC-25 DECLARED DEFECTIVE, FEBRUARY 1987

On February 10, 1987, the FDA declared the Therac-25 to be defective. In order for the Therac-25 to gain back FDA approval, AECL had to demonstrate how it would make the system safe. Five months later, after five revisions, AECL produced a corrective action plan that met the approval of the FDA. This plan incorporated a variety of hardware interlocks to prevent the machine from delivering overdoses or activating the beam when the turntable was not in the correct position.

8.5.3 Software Errors

In the course of investigating the accidents, AECL discovered a variety of hardware and software problems with the Therac-25. Two of the software errors are examples of race conditions. In a **race condition,** two or more concurrent tasks share a variable, and the order in which they read or write the value of the variable can affect the behavior of the program. Race conditions are extremely difficult to identify and fix, because usually the two tasks do not interfere with each other and nothing goes wrong. Only in rare conditions will the tasks actually interfere with each other as they manipulate the variable, causing the error to occur. We describe both of these errors to give you some insight into how difficult they are to detect.

The accidents at the ETCC occurred because of a race condition associated with the command screen (Figure 8.5). One task was responsible for handling keyboard input and making changes to the command screen. A second task was responsible for monitoring the command screen for changes and moving the magnets into position. After the operator uses the first task to complete the prescription (1), the second task sees the cursor in the lower right-hand corner of the screen and begins the eight-second process of moving the magnets (2). Meanwhile, the operator sees her mistake. The first task responds to her keystrokes and lets her change the "X" to an "E" (3). She gets the cursor back to the lower right-hand corner before eight seconds are up (4). Now the second task finishes moving the magnets (5). It sees the cursor in the lower right-hand corner of the screen and incorrectly assumes the screen has not changed. The crucial substitution of electron beam for x-ray goes unnoticed.

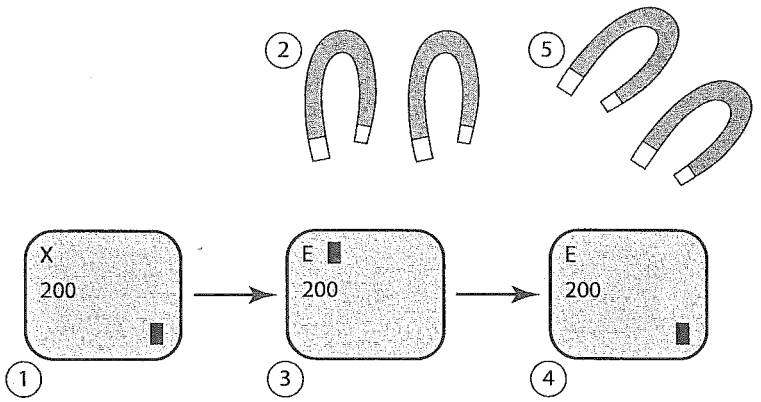

FIGURE 8.5 Illustration of a Therac-25 bug revealed by fast-typing operators. (1) The operator finishes filling in the form. The software knows the form is filled in because the cursor is in the lower right-hand corner of the screen. (2) The software instructs the magnets to move into the correct positions. While the magnets are moving, the software does not check for screen edits. (3) The operator changes the prescription from x-ray to electron beam. (4) The operator finishes the edit, returning the cursor to the lower right-hand corner of the screen. (5) The magnets finish moving. The software now checks the screen cursor. Since it is in the lower right-hand corner, the program assumes there have been no edits.

What makes this bug particularly treacherous is that it only occurs with faster, more experienced operators. Slower operators would not be able to complete the edit and get the cursor back to the lower right-hand corner of the screen in only eight seconds. If the cursor happened to be anywhere else on the screen when the magnets stopped moving, the software would check for a screen edit and there would be no overdose. It is ironic that the safety of the system actually *decreased* as the experience of the operator *increased*.

Another race condition was responsible for the overdoses at the Yakima Valley Memorial Hospital (Figure 8.6). It occurred when the machine was putting the electron-beam gun back into position. A variable was supposed to be 0 if the gun was ready to fire. Any other value meant the gun was not ready. As long as the electron beam gun was out of position, one task kept incrementing that variable. Unfortunately, the variable could only store the values from 0 to 255. Incrementing it when it had the value 255 would result in the variable's value rolling over to 0, like a car's odometer.

Nearly every time that the operator hit the SET button when the gun was out of position, the variable was not 0 and the gun did not fire (Figure 8.6a). However, there was a very slight chance that the variable would have just rolled over when the operator hit the SET button (Figure 8.6b). In this case the accelerator would emit a charge, even though the system was not ready.

8.5.4 Postmortem

Let's consider some of the mistakes AECL made in the design, development, and support of this system.

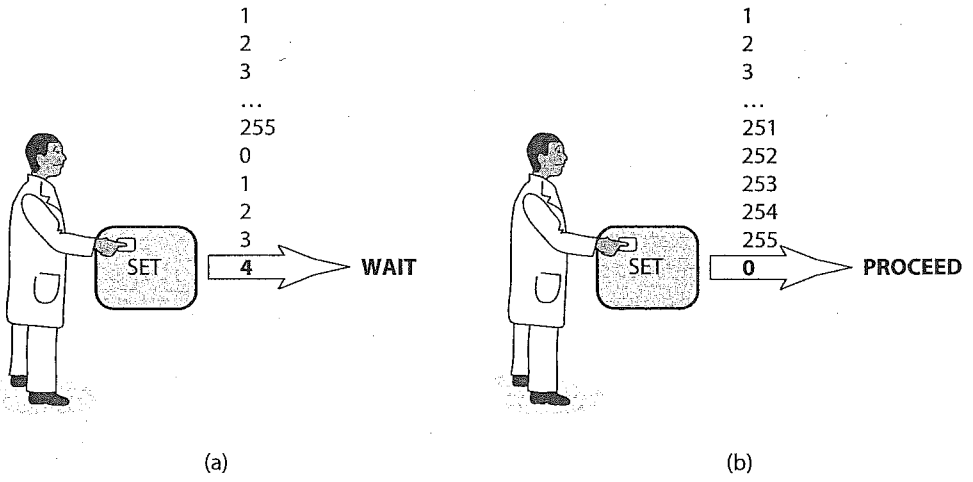

FIGURE 8.6 The Therac-25 could administer radiation too soon if the operator hit the SET button at precisely the wrong time. As long as the electron-beam gun was out of position, a software task kept incrementing an 8-bit variable. (a) Usually when the operator hit the SET button, the variable was not zero and the system would wait, just as it was supposed to. (b) If the operator hit the SET button just as the variable rolled over from 255 to 0, the system would administer radiation, even though the gun was out of position.

When accidents were reported, AECL focused on identifying and fixing particular software bugs. This approach was too narrow. As Nancy Leveson and Clark Turner point out, "most accidents are system accidents; that is, they stem from complex interactions between various components and activities" [47]. The entire system was broken, not just the software. A strategy of eliminating bugs assumes that at some point the last bug will be eradicated. But as Leveson and Turner write, "There is always another software bug" [47].

The real problem was that the system was not designed to be fail-safe. Good engineering practice dictates that a system should be designed so that no single point of failure will lead to a catastrophe. By relying completely upon software for protection against overdoses, the Therac-25 designers ignored this fundamental engineering principle.

Another flaw in the design of the Therac-25 was its lack of software or hardware devices to detect and report overdoses and shut down the accelerator immediately. Instead, the Therac-25 designers left it up to the patients to report when they had received overdoses.

There are also particular software lessons we can learn from the case of the Therac-25. First, it is very difficult to find software errors in programs where multiple tasks execute at the same time and interact through shared variables. Second, the software design needs to be as simple as possible, and design decisions must be documented to aid in the maintenance of the system. Third, the code must be reasonably documented at the time it is written.

Fourth, reusing code does not always increase the quality of the final product. AECL assumed that by reusing code from the Therac-6 and Therac-20, the software would be more reliable. After all, the code had been part of systems used by customers for years with no problems. This assumption turned out to be wrong. The earlier codes did contain errors. These errors remained undetected because the earlier machines had hardware interlocks that prevented the computer's erroneous commands from harming patients.

The tragedy was compounded because AECL did not communicate fully with its customers. For example, AECL told the physicists in Washington and Texas that an overdose was impossible, even though AECL had already been sued by the patient in Georgia.

8.5.5 Moral Responsibility of the Therac-25 Team

Should the developers and managers at AECL be held morally responsible for the deaths resulting from the use of the Therac-25 they produced?

In order for a moral agent to be responsible for a harmful event, two conditions must hold:

- *Causal condition*: the actions (or inactions) of the agent must have caused the harm.
- *Mental condition*: the actions (or inactions) must have been intended or willed by the agent.

In this case, the causal condition is easy to establish. The deaths resulted both from the action of AECL employees (creating the therapy machine that administered the overdose) and the inaction of AECL employees (failing to withdraw the machine from service or even inform other users of the machine that there had been overdoses).

What about the second condition? Surely the engineers at AECL did not intend or try to create a machine that would administer lethal overdoses of radiation. However, philosophers also extend the mental condition to include unintended harm if the moral agent's actions were the result of carelessness, recklessness, or negligence. The design team took a number of actions that fall into this category. It constructed a system without hardware interlocks to prevent overdoses or the beam from being activated when the turntable was not in a correct position. The machine had no software or hardware devices to detect an accidental overdose. Management allowed software to be developed without adequate documentation. It presumed the correctness of reused code and failed to test it thoroughly. For these reasons, the mental condition holds as well, and we conclude the Therac-25 team at AECL is morally responsible for the deaths caused by the Therac-25 radiation therapy machine.

8.5.6 Postscript

More than two decades after the Therac-25 accidents, computer errors related to radiation machines continue to maim and kill patients. In late 2006, Scott Jerome-Parks received three overdoses from a linear accelerator at a New York City Hospital that led to his death a few weeks later. He was only 43 years old. About the same time, 32-year-old breast cancer patient Alexandra Jn-Charles received 27 straight days of radiation overdoses at another New York hospital that led to her death. An investigation of radiation overdoses by *The New York Times* concluded that a variety of errors, including faulty software, were leading to crippling or fatal accidents [48].

8.6 Computer Simulations

In the previous section, we focused on an unreliable computer-controlled system that delivered lethal doses of radiation to cancer patients, but even systems kept behind the locked doors of a computer room can cause harm. Errors in computer simulations can result in poorly designed products, mediocre science, and bad policy decisions. In this section, we review our growing reliance on computer simulations for designing products, understanding our world, and even predicting the future, and we describe ways in which computer modelers validate their simulations.

8.6.1 Uses of Simulation

Computer simulation plays a key role in contemporary science and engineering. There are many reasons why a scientist or engineer may not be able to perform a physical experiment. It may be too expensive or time-consuming, or it may be unethical or impossible to perform. Computer simulations have been used to design nuclear weapons, search

Here is even blunter language from the license agreement accompanying Harmonic Visions's Music Ace program:

> WE DO NOT WARRANT THAT THIS SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE. WE EXCLUDE AND EXPRESSLY DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES NOT STATED HEREIN, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

In other words, don't blame us if the program doesn't do what you hoped it would do, or if it crashes all the time, or if it is full of bugs.

8.8.2 Are Software Warranties Enforceable?

How can software manufacturers get away with disclaiming any warranties on their products? It's not clear that they can. Article 2 of the Uniform Commercial Code (UCC) governs the sale of products in the United States. In 1975, Congress passed the Magnuson-Moss Warranty Act. One goal of the act was to prevent manufacturers from putting unfair warranties on products costing more than $25. A second goal was to make it economically feasible for consumers to bring warranty suits by allowing courts to award attorneys' fees. Together, the Magnuson-Moss Warranty Act and Article 2 of the UCC protect the rights of consumers. A computer program is a product. Hence, unfair warranties on shrinkwrap software could be in violation of these laws.

An early court case, *Step-Saver Data Systems v. Wyse Technology and The Software Link*, seemed to affirm the notion that software manufacturers could be held responsible for defects in their products, despite what they put in their warranties. However, two later cases seemed to indicate the opposite. In *ProCD v. Zeidenberg*, the court ruled that the customer was bound to the license agreement, even if the license agreement does not appear on the outside of the shrinkwrap box. *Mortenson v. Timberline Software* showed that a warranty disclaiming the manufacturer's liability could hold up in court.

STEP-SAVER DATA SYSTEMS V. WYSE TECHNOLOGY AND THE SOFTWARE LINK

Step-Saver Data Systems, Inc. sold timesharing computer systems consisting of an IBM PC AT server, Wyse terminals, and an operating system provided by The Software Link, Inc. (TSL). In 1986–1987, Step-Saver purchased and resold 142 copies of the Multilink Advanced operating system provided by TSL.

To purchase the software, Step-Saver called TSL and placed an order, then followed up with a purchase order. According to Step-Saver, the TSL phone sales representatives said that Multilink was compatible with most DOS applications. The box containing the Multilink software included a licensing agreement in which TSL disclaimed all express and implied warranties.

Step-Saver's timesharing systems did not work properly, and the combined efforts of Step-Saver, Wyse, and TSL could not fix the problems. Step-Saver was sued by twelve of its customers. In turn, Step-Saver sued Wyse Technology and TSL.

The Third Circuit of the U.S. Court of Appeals ruled in favor of Step-Saver [57]. It based its argument on Article 2 of the UCC. The court held that the original contract between Step-Saver and TSL consisted of the purchase order, the invoice, and the oral statements made by TSL representatives on the telephone. The license agreement had additional terms that would have materially altered the contract. However, Step-Saver never agreed to these terms.

The court wrote, "In the absence of a party's express assent to the additional or different terms of the writing, section 2-207 [of the UCC] provides a default rule that the parties intended, as the terms of their agreement, those terms to which both parties have agreed along with any terms implied by the provision of the UCC." The court noted that the president of Step-Saver had objected to the terms of the licensing agreement. He had refused to sign a document formalizing the licensing agreement. Even after this, TSL had continued to sell to Step-Saver, implying that TSL wanted the business even if the contract did not include the language in the licensing agreement. That is why the court ruled that the purchase order, the invoice, and the oral statements constituted the contract, not the license agreement.

PROCD, INC. V. ZEIDENBERG

ProCD invested more than $10 million to construct a computer database containing information from more than 3,000 telephone directories. ProCD also developed a proprietary technology to compress and encrypt the data. It created an application program enabling users to search the database for records matching criteria they specified. ProCD targeted its product, called SelectPhone, to two different markets: companies interested in generating mailing lists, and individuals interested in finding the phone numbers or addresses of particular people they wanted to call or write. Consumers who wanted SelectPhone for personal use could purchase it for $150; companies paid much more for the right to put the package to commercial use. ProCD included in the consumer version of SelectPhone a license prohibiting the commercial use of the database and program. In addition, the license terms were displayed on the user's computer monitor every time the program was executed.

Matthew Zeidenberg purchased the consumer version of SelectPhone in 1994. He formed a company called Silken Mountain Web Services, Inc., which resold the information in the SelectPhone database. The price it charged was substantially less than the commercial price of SelectPhone. ProCD sued Matthew Zeidenberg for violating the licensing agreement.

At the trial, the defense argued that Zeidenberg could not be held to the terms of the licensing agreement, since they were not printed on the outside of the box containing the software. The U.S. Court of Appeals for the Seventh Circuit ruled in favor of ProCD. Judge Frank Easterbrook wrote, "Shrinkwrap licenses are enforceable unless their terms are objectionable on grounds applicable to contracts in general (for example, if they violate a rule of positive law, or if they are unconscionable)" [58].

MORTENSON V. TIMBERLINE SOFTWARE

M. A. Mortenson Company was a national construction contractor with a regional office in Bellevue, Washington. Timberline Software, Inc. produced software for the construction industry. Mortenson had used software from Timberline for several years. In July 1993, Mortenson purchased eight copies of a bidding package called Precision Bid Analysis.

Timberline's licensing agreement included this paragraph:

> LIMITATION OF REMEDIES AND LIABILITY.
>
> NEITHER TIMBERLINE NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THE PROGRAMS OR USER MANUALS SHALL BE LIABLE TO YOU FOR ANY DAMAGES OF ANY TIME, INCLUDING BUT NOT LIMITED TO, ANY LOST PROFITS, LOST SAVINGS, LOSS OF ANTICIPATED BENEFITS, OR OTHER INCIDENTAL, OR CONSEQUENTIAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAMS, WHETHER ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT TORT, OR UNDER ANY WARRANTY, OR OTHERWISE, EVEN IF TIMBERLINE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR FOR ANY OTHER CLAIM BY ANY OTHER PARTY. TIMBERLINE'S LIABILITY FOR DAMAGES IN NO EVENT SHALL EXCEED THE LICENSE FEE PAID FOR THE RIGHT TO USE THE PROGRAMS.

In December 1993, Mortenson used Precision Bid Analysis to prepare a bid for the Harborview Medical Center in Seattle. On the day the bid was due, the software malfunctioned. It printed the message "Abort: Cannot find alternate" 19 times. Mortenson continued to use the software and submitted the bid the software produced. After the firm won the contract, Mortenson discovered that its bid was $1.95 million too low.

Mortenson sued Timberline for breach of express and implied warranties. It turns out Timberline had been aware of the bug uncovered by Mortenson since May 1993. Timberline had fixed the bug and already sent a newer version of the program to some of its other customers who had encountered it. It had not sent the improved program to Mortenson. Nevertheless, Timberline argued that the lawsuit be summarily dismissed because the licensing agreement limited the consequential damages that Mortsenson could recover from Timberline. The King County Superior Court ruled in favor of Timberline. The ruling was upheld by the Washington Court of Appeals and the Supreme Court of the State of Washington [59].

8.8.3 Moral Responsibility of Software Manufacturers

Should producers of shrinkwrap software be held responsible for defects in their programs?

Let's consider the consequences of holding manufacturers of shrinkwrap software liable for damages, such as lost profits, caused by errors encountered by licensees. Currently, manufacturers rely upon consumers to help them identify bugs in their products. If they must find these bugs themselves, they will need to hire many more software testers. The result will be higher prices and longer program development times.