extending or improving the program can do so. A "benevolent dictator" who keeps track of what is going on governs most open-source projects. If a change or improvement passes the peer review of fellow developers and gets incorporated in the next version, it is a great coup.

Linux is the best known open source project. Linus Torvolds wrote the first simple version of the operating system using Unix as a blueprint and continued to oversee its development. IBM spent $1 billion on Linux in 2001 with the object of making it a computing standard. As *The Economist* says,

> "Some people like to dismiss Linux as nothing more than a happy accident, but the program looks more like a textbook example of an emerging pattern.... Open source is a mass phenomenon, with tens of thousands of volunteer programmers across the world already taking part, and more joining in all the time, particularly in countries such as China and India. SourceForge, a web site for developers, now hosts more than 18,000 open-source projects that keep 145,000 programmers busy."

Only time will tell if the open-source software development movement contributes to producing more high-quality products.

Notorious Software Errors

Everyone involved in computing has his or her favorite software horror story. We include only a small sample here.

AT&T Down for Nine Hours

In January of 1990, AT&T's long-distance telephone network came to a screeching halt for nine hours, because of a software error in the electronic switching systems. Of the 148 million long-distance and 800-number calls placed with AT&T that day, only 50% got through. This failure caused untold collateral damage:

- Hotels lost bookings.
- Rental car agencies lost rentals.
- American Airlines' reservation system traffic fell by two-thirds.
- A telemarketing company lost $75,000 in estimated sales.
- MasterCard didn't get to process its typical 200,000 credit approvals.
- AT&T lost some $60 to $75 million.

As AT&T Chairman Robert Allen said, "It was the worst nightmare I've had in 32 years in the business." [11]

How did this happen? Earlier versions of the switching software worked correctly. The software error was in the code that upgraded the system to make it respond more quickly to a malfunctioning switch. The error involved a *break* statement in the C-code.[12] As Henry Walker points out in *The Limits of Computing*, this breakdown illustrates several points common to many software failures. The software had been tested extensively before its release, and it worked correctly for about a month. In addition to testing, code reviews had been conducted during development. One programmer made the error, but many others reviewed the code without noticing the error. The failure was triggered by a relatively uncommon sequence of events, difficult to anticipate in advance. And the error occurred in code designed to improve a correctly working system; that is, during the maintenance phase. E. N. Adams in the *IBM Journal of Research and Development* estimates that 15 to 50% of attempts to remove an error from a large program result in the introduction of additional errors.

Therac-25

One of the most widely cited software-related accidents involved a computerized radiation therapy machine called the Therac-25. Between June 1985 and January 1987, six known accidents involved massive overdoses by the Therac-25, leading to deaths and serious injuries. These accidents have been described as the worst series of radiation accidents in the 35-year history of medical accelerators.

It is beyond the scope of this book to go into a detailed analysis of the software failure. Suffice it to say there was only a single coding error, but tracking down the error exposed that the whole design was seriously flawed. Leveson and Turner in their article in *IEEE Computer*, add this scathing comment:

> "A lesson to be learned from the Therac-25 story is that focusing on particular software bugs is not the way to make a safe system. Virtually all complex software can be made to behave in an unexpected fashion under certain conditions. The basic mistakes here involved poor software-engineering practices and building a machine that relies on the software for safe operation. Furthermore, the particular coding error is not as important as the general unsafe design of the software overall." [13]

Bugs in Government Projects

On February 25, 1991, during the Gulf War, a Scud missile struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. An American Patriot Missile battery in Dhahran, Saudi Arabia, failed to track and intercept the incoming Iraqi Scud missile because of a software error. This error, however, was not a coding error

but a design error. A calculation involved a multiplication by 1/10, which is a non-terminating number in binary. The resulting arithmetic error accumulated over the 100 hours of the batteries' operation amounted to .34 seconds, enough for the missile to miss its target.[14]

The General Accounting Office concluded:

> "The Patriot had never before been used to defend against Scud missiles nor was it expected to operate continuously for long periods of time. Two weeks before the incident, Army officials received Israeli data indicating some loss in accuracy after the system had been running for 8 consecutive hours. Consequently, Army officials modified the software to improve the system's accuracy. However, the modified software did not reach Dhahran until February 26, 1991—the day after the Scud incident." [15]

The Gemini V missed its expected landing point by about 100 miles. The reason? The design of the guidance system did not take into account the need to compensate for the motion of Earth around the Sun.[16]

In October 1999, the Mars Climate Orbiter entered the Martian atmosphere about 100 kilometers lower than expected, causing the craft to burn up. Arthur Stephenson, chairman of the Mars Climate Orbiter Mission Failure Investigation Board concluded:

> "The 'root cause' of the loss of the spacecraft was the failed translation of English units into metric units in a segment of ground-based, navigation-related mission software, as NASA has previously announced … The failure review board has identified other significant factors that allowed this error to be born, and then let it linger and propagate to the point where it resulted in a major error in our understanding of the spacecraft's path as it approached Mars." [17]

Launched in July of 1962, the Mariner 1 Venus probe veered off course almost immediately and had to be destroyed. The problem was traced to the following line of Fortran code:

```
I ! L " eP 1
```

The period should have been a comma. An $18.5 million space exploration vehicle was lost because of this typographical error.

17.3 Problems

There are problems for which it is easy to develop and implement computer solutions. There are problems for which we can implement computer solutions, but we wouldn't get the results in our lifetime. There