The New York Times
CyberTimes

December 1, 1996

FAST FORWARD / JAMES GLEICK

1 Teensy Little Bug, 1 Humongous Crash

It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

Illustration: John Hersey

All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.

One bug, one crash. Of all the careless lines of code recorded in the annals of computer science, this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.

To play the tape backward:

At 39 seconds after launch, as the rocket reached an altitude of two and a half miles, a self-destruct mechanism finished off Ariane 5, along with its payload of four expensive and uninsured scientific satellites. Self-destruction was triggered automatically because aerodynamic forces were ripping the boosters from the rocket.

This disintegration had begun instantaneously when the spacecraft swerved off course under the pressure of the three powerful nozzles in its boosters and main engine. The rocket was making an abrupt course correction that was not needed, compensating for a wrong turn that had not taken place.

Steering was controlled by the on-board

Investigators did not attempt to calculate how much time or money was saved by omitting the standard error-protection code.

Steering was controlled by the on-board computer, which mistakenly thought the rocket needed a course change because of numbers coming from the inertial guidance system. That device uses gyroscopes and accelerometers to track motion. The numbers looked like flight data -- bizarre and impossible flight data -- but were actually a diagnostic error message. The guidance system had in fact shut down.

This shutdown occurred 36.7 seconds after launch, when the guidance system's own computer tried to convert one piece of data -- the sideways velocity of the rocket -- from a 64-bit format to a 16-bit format. The number was too big, and an overflow error resulted.

When the guidance system shut down, it passed control to an identical, redundant unit, which was there to provide backup in case of just such a failure. But the second unit had failed in the identical manner a few milliseconds before. It was running the same software.

This bug belongs to a species that has existed since the first computer programmers realized they could store numbers as sequences of bits, atoms of data, ones and zeroes: 1001010001101001. . . . A bug like this might crash a spreadsheet or word processor on a bad day. Ordinarily, though, when a program converts data from one form to another, the conversions are protected by extra lines of code that watch for errors and recover gracefully. Indeed, many of the data conversions in the guidance system's programming included such protection.

But in this case, the programmers had decided that this particular velocity figure would never be large enough to cause trouble. After all, it never had been before. Unluckily, Ariane 5 was a faster rocket than Ariane 4. One extra absurdity: the calculation containing the bug, which shut down the guidance system, which confused the on-board computer, which forced the rocket off course, actually served no purpose once the rocket was in the air. Its only function was to align the system before launch. So it should have been turned off. But engineers chose long ago, in an earlier version of the Ariane, to leave this function running for the first 40 seconds of flight -- a "special feature" meant to make it easy to restart the system in the event of a brief hold in the countdown.

When a routine program turns into rocket science, bugs are

The Europeans hope to launch a new Ariane 5 next spring, this time with a newly designated "software architect" who will oversee a process of more intensive and, they hope, realistic ground simulation. Simulation is the great hope of software debuggers everywhere, though it can never anticipate every feature of real life. "Very

magnified a thousandfold.

tiny details can have terrible consequences," says Jacques Durand, head of the project, in Paris. "That's not surprising, especially in a complex software system such as this is."

These days, we have complex software systems everywhere. We have them in our dishwashers and in our wristwatches, though they're not quite so mission-critical. We have computers in our cars -- from 15 to 50 microprocessors, depending how you count: in the engine, the transmission, the suspensions, the steering, the brakes and every other major subsystem. Each runs its own software, thoroughly tested, simulated and debugged, no doubt.

Bill Powers, vice president for research at Ford, says that cars' computing power is increasingly devoted not just to actual control but to diagnostics and contingency planning -- "Should I abort the mission, and if I abort, where would I go?" he says. "We also have what's called a limp-home strategy." That is, in the worst case, the car is supposed to behave more or less normally, like a car of the pre-computer era, instead of, say, taking it upon itself to swerve into the nearest tree.

The European investigators chose not to single out any particular contractor or department for blame. "A decision was taken," they wrote. "It was not analyzed or fully understood." And "the possible implications of allowing it to continue to function during flight were not realized." They did not attempt to calculate how much time or money was saved by omitting the standard error-protection code.

"The board wishes to point out," they added, with the magnificent blandness of many official accident reports, "that software is an expression of a highly detailed design and does not fail in the same sense as a mechanical system." No. It fails in a different sense. Software built up over years from millions of lines of code, branching and unfolding and intertwining, comes to behave more like an organism than a machine.

"There is no life today without software," says Frank Lanza, an executive vice president of the American rocket maker Lockheed Martin. "The world would probably just collapse." Fortunately, he points out, really important software has a reliability of 99.9999999 percent. At least, until it doesn't.

You are invited to read other columns and articles by James Gleick by visiting his home page at http://www.around.com.