the Availability Digest

www.availabilitydigest.com
@availabilitydig

Failsafe

August 2016

I recently watched the 1964 movie thriller “Failsafe.” Back then, the United States kept multiple strategic bombers armed with nuclear bombs in the air at all times to deter a nuclear attack from the Soviet Union. If a suspected enemy incursion was detected by radar, the bombers would all head for Russia until they arrived at the “failsafe” point. If at that point they were not called back, they would continue to Russia to drop their nuclear weapons. Once past the failsafe point, they were to ignore any other messages in order to prevent the Soviet Union from faking recall messages.

During one such suspected incursion, the Strategic Air Command determined that the radar target was an airliner that had been blown off course by high tail winds and a failed engine. The U.S. bombers were called back, but one bomber group did not receive the recall directive. The recall directive was jammed by the Soviet Union in a test of a new capability it had recently developed. The bomber group continued on a course to Moscow. The movie focuses on the efforts, all unsuccessful, to recall the errant bomber group.

During the attempts to recall the bomber, at minute 54 in the movie, a Mr. Knapp of Amalgamated Electronics makes the statement:

> “The more complex an electronic system gets, the more accident prone it is. Sooner or later, it breaks down. Machines we make are so fast, so accurate, and the mistakes they make are so subtle, very often humans just can’t know whether the machine is lying or telling the truth.”

Fifty years later, this is still true, though now we incorporate redundant systems in an attempt to ensure that we are “getting the truth.” Back in 1964, computers were too big and too expensive for redundancy.

This takes me back to my early days in computer development. As I was going for my Master’s degree in Electrical Engineering at MIT, the college was using Whirlwind, a vacuum-tube computer it had developed to compute projectile trajectories. Programs could take hours or even days to run. However, Whirlwind’s MTBF was about eight hours, so each program had to store its results every few hours so that it could restart following a failure.

As an MIT research assistant, I was assigned to Lincoln Laboratories, which was developing the first transistorized computer, the TX0, for the U.S. Air Force. We would design a major component of the computer, such as a register, and lay out a full circuit board for it (often measuring a foot or more). The circuit board frequently did not work, and a new one would be created and tested. Progress was very slow. Furthermore, transistors came as a single package, were very expensive, and were very sensitive. It was easy to burn one out. Whenever that happened, we had to write a complete report on how the destruction of the transistor happened.

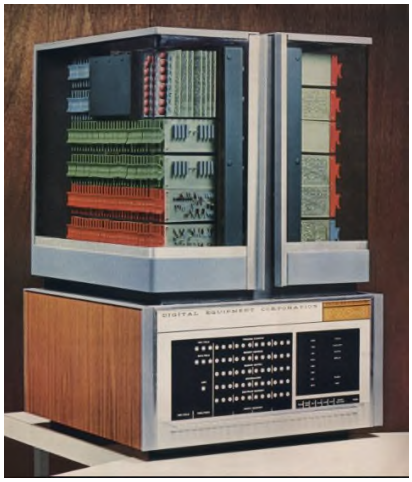
The supervisor of my group came up with a great idea. Rather than laying out circuit boards that could not be changed easily, he developed a set of plug-in modules for different types of logic units. The modules looked like small vacuum tubes and each held a flip-flop, a gate, or some other logical unit. The modules

plugged into a chassis much like vacuum tubes, and they were wired into circuits using a simple wire-wrap tool on the back of the chassis. Suddenly, it was easy to design and build a major computer component. Transistors were not destroyed. Progress on TX0 increased significantly.

As an aside, my supervisor decided to start a company building computers with this technology. His name was Ken Olsen, and his company was Digital Equipment Corp. His line of computers was the PDP series, and they ran on the company's VMS operating system (which later became the OpenVMS operating system). The logic modules were converted to printed-circuit cards called "flip chips."

Digital ultimately was bought by Compaq, which was acquired by HP. HP spun off the Digital computer system, now known as OpenVMS, to a new company that was set up specifically to market Digital computers. The company, VMS Software, Inc., actively develops, markets, and supports OpenVMS systems.

My favorite computer was the PDP-8. In terms of numbers sold, it was the most successful minicomputer of its time, with over 50,000 systems sold by Digital. It used a twelve-bit word size, and the basic system came with 4K of memory (yes, K, not G). Why a twelve-bit word size? Eight-bit bytes had not been invented yet, and characters were represented by six-bit codes.



A Digital PDP-8 Minicomputer

The PDP-8 came with no operating system except for device drivers. I implemented a COBOL-like language for it called SAIBOL-8 (SAI Business-Oriented language for the PDP-8 – my company was Sombers Associates, Inc., or SAI). Using SAIBOL-8, I developed a payroll package that could process the payrolls of up to 50 small companies at the same time.

I started a payroll company called MiniData to process payrolls for small companies. Our billboards read, "You pay us $10. We'll pay 15," implying that we would do a fifteen-person payroll for $10. My "data center" (a small room) held two PDP-8s for redundancy. Each had 8K of memory, 32 K of disk, and small Digital magnetic tape units. The lower 4K of memory was used to hold SAIBOL-8 and the device drivers. Application programs were paged into the upper 4K of memory as needed. Can you imagine today creating any kind of usable software package on a machine with only 8K of memory?

Our first payrolls proved Mr. Knapp's prediction. The payroll checks were totally wrong – the machine was definitely lying. Our software was loaded with bugs. For a week, we did payrolls by hand until we got the programs correct. The machine then began telling the truth.

MiniData took off like a shot. We did payrolls for hundreds of companies, and several years later MiniData was acquired for $5 million.

But Mr. Knapp's statement still held. Could you ever know whether the machine was lying or telling the truth? Computers are complex machines and can malfunction easily in ways that are not obvious.

In response to this problem, several companies were formed to design and market fault-tolerant computers.[1] The main survivors were Tandem and Stratus systems.

Tandem systems were born in the late 1970s. Each system contains from two to sixteen processors, and each program running in one processor has a backup in another processor. Should a processor fail, the backup program takes over in subsecond time and continues processing. Users are unaware of the failure.

[1] The Dawn of Fault-Tolerant Computing, *Availability Digest*; April 2016.
http://www.availabilitydigest.com/public_articles/1104/survivable_systems.pdf

The Stratus system was not only fault-tolerant but also failsafe. It comprised two processors running in lockstep. The processors compared their results before any memory write. If there was a mismatch, the processors entered a fault-resolution mode and resolved the problem. Therefore, the results delivered by the Stratus system were always guaranteed to be correct (at least, to the extent that the software was correct).

If a computer lies, it is most likely a software problem. Software always has bugs that are fixed over time. However, it is likely that even after extended times, left in the software will be latent bugs that can cause it to "lie." We depend heavily in our daily lives on computers, and eliminating these bugs so that the computer will always tell the truth is an ongoing process.

Summary

It is certainly true that the more complex a system is, the more likely it is that it will provide erroneous outputs. Computers are a major class of this sort of system. Because their actions are governed by the humans who program them, and because humans are not fail-safe, neither are computers. Though the movie "Failsafe" was not about computers (mid-1964 was too early for that), it emphasized the frailty of humans in complex situations.