http://articles.latimes.com/2010/mar/11/opinion/la-oew-cummings12-2010mar12

Haven't found that software glitch, Toyota? Keep trying

An electronics problem isn't to blame for the sudden acceleration, say the carmaker's engineers. That's nearly impossible to conclusively determine through laboratory tests.

March 11, 2010

By David M. Cummings

There has been a lot of speculation recently that Toyota's problems with sudden acceleration may be caused by a problem in the vehicles' electronics systems. The "electronics" includes millions of lines of software running on the automobiles' computers. As The Times reported on March 3, Toyota's chief engineer testified to Congress that the company has done extensive testing on its cars' electronics and believes they are not the cause of the sudden acceleration.

Having owned a Toyota myself, I have always been a fan of what I perceived to be the automaker's high standards for quality. I also happen to have more than three decades of experience designing, building and researching reliable computer systems, many of which are embedded inside other devices. Based on this experience, I find it very difficult to accept the statements from Toyota's chief engineer. And the implications extend beyond Toyota, to all other companies that rely on software for their product safety.

As anyone with experience in embedded systems will tell you, there are nasty software bugs that can be extremely difficult to reproduce in a laboratory test environment. To illustrate, I'd like to describe one such bug we encountered at the Jet Propulsion Laboratory while developing the flight software for NASA's Mars Pathfinder spacecraft.

Because of Pathfinder's high reliability requirements and the probability of unpredictable hardware errors due to the increased radiation effects in space, we adopted a highly "defensive" programming style. This included performing extensive error checks in the software to detect the possible side effects of radiation-induced hardware glitches and certain software bugs.

One member of our team, Steve Stolper, had a simple arithmetic computation in his software that was guaranteed to produce an even result (2, 4, 6 and so on) if the computer was working correctly. Many programmers would not bother to check the result of such a simple computation. Stolper, however, put in an explicit test to see if the result was even. We referred to this test as his "two-plus-two-equals-five check." We never expected to see it fail.

Lo and behold, during software testing we saw Stolper's error message indicating the check had failed. We saw it just once. We were never able to reproduce the failure, despite repeated attempts over many thousands if not millions of iterations. We scratched our heads. How could this happen, especially in the benign environment of our software test lab, where radiation effects were virtually nonexistent? We looked carefully at Stolper's code, and it was sound.

The only viable theory we could come up with was that an interrupt (an external hardware stimulus such as a timer going off) had occurred at just the right microsecond within the execution of Stolper's software. Furthermore, we theorized, the operating system (the equivalent of Windows on the flight computer) had a bug that caused it to misremember whether an arithmetic carry had occurred just before the interrupt. Although highly unlikely, it was the only credible explanation we could come up with. Because this was a new version of the operating system built for Pathfinder, still not yet fully tested itself, this theory had some credibility.

We reviewed the operating system code and consulted with the company that developed it. Much to our surprise (and relief), we found that there was indeed a bug in the interrupt handling software as we had theorized. If Stolper had not put in his "two-plus-two-equals-five check," we might not have found the problem until it was too late -- that is, until it caused a catastrophic error en route to Mars. (There were other subtle bugs that we found and fixed before and after launch, some arguably subtler than this one.)

So what's my point? First, I don't know if Toyota's engineers embrace the software reliability approaches we embraced on Pathfinder, which allowed us to catch these subtle bugs. Second, even if the Toyota engineers do everything we did on Pathfinder and more, I'm still skeptical when I hear an engineer declare a complex software system to be bug-free based on laboratory testing. It is extremely difficult to make such a determination through laboratory tests. I'm quite certain none of the members of the Pathfinder software team would have declared the software to be bug-free prior to launch, despite our best efforts to make it so.

If Toyota has indeed tested its software as thoroughly as it says without finding any bugs, my response is simple: Keep trying. Find new ways to instrument the software, and come up with more creative tests. The odds are that there are still bugs in the code, which may or may not be related to unintended acceleration. Until these bugs are identified, how can you be certain they are not related to sudden acceleration?

My last point is this: Whatever the final outcome of the Toyota saga, this should serve as a wake-up call to all industries that increasingly rely on software for safety. It is probably only a matter of time before a software error results in injury or death, if it has not happened already (there are some who say it has). We need to minimize that possibility by enforcing extremely stringent standards on the development and testing of software in all safety-critical systems, including, but not limited to, automobiles.

David M. Cummings, executive vice president of the Santa Barbara-based Kelly Technology Group, spent nine years as a consultant for the Jet Propulsion Laboratory, where he worked on the Mars Pathfinder spacecraft.