The Worst Computer Bugs in History: Race conditions in Therac-25

COMMUNITY

The Worst Computer Bugs in History: Race conditions in Therac-25

JAMIE LYNCH
September 19th, 2017

> The Worst Computer Bugs in History is a mini series to commemorate the discovery of the first computer bug seventy years ago. Although these stories are more extreme than most software bugs engineers will encounter during their careers, they are worth studying for the insights they can offer into software development and deployment. These computer bugs left a significant impact on the people who experienced them, and we hope they'll offer valuable lessons we can all apply to our own work and projects. Read about other computer bugs in the series: The Ariane 5 Disaster, the Mars Climate Orbiter, and losing $460 million in 45 minutes.

Therac-25 causes radiation overdoses

The Therac-25 was a radiation therapy machine manufactured by AECL in the 80s, which offered a revolutionary dual treatment mode. It was also designed from the outset to use software based safety systems rather than hardware controls.

The removal of these hardware safety measures had tragic consequences, as race conditions in the codebase led to the death of three patients, and caused debilitating injuries to at least three other patients. The manufacturer ultimately became the target of several lawsuits from families of the victims, and became subject to a Class I recall from the FDA, a situation which only happens if the agency believes there is significant risk of death or serious injury through continued use of a medical device.

What is radiation?

Radiation is toxic to living cells, and in large enough quantities will cause them to die. For example, cancerous cells can be killed by a dose of radiation which specifically targets the affected area. However, care must be taken to ensure that the dose is not too large, as this would kill healthy surrounding tissue, and could lead to serious injury or death for the patient.

The Therac-25 was a linear accelerator with two modes of operation. Firstly, it could fire a beam of low-energy electrons, which do not penetrate far into the body, and are therefore well-suited at killing shallow tissues, such as in skin cancer. The second mode of operation delivers radiation via a beam of higher-energy X-Ray photons. These particles travel further and are best suited to treating deeper tissues, such as cancer of the lungs.

Revolutionary Design

Dual-mode operation was truly revolutionary at the time. Hospitals would not need to maintain two separate machines, reducing their maintenance costs, and logistics could be simplified, as patients would not need to be moved from one room to the next.

The low-power mode used scanning magnets to spread the electron beam, whereas the high-power mode was activated by rotating four components into the beam. This process took around 8 seconds to complete, and afterwards, it would spread and direct a beam of the appropriate strength towards its target.

Software, not hardware, for safety controls

Therac-25 relied on software controls to switch between modes, rather than physical hardware. Preceding models used separate circuits to monitor radiation intensity, and hardware interlocks to ensure that spreading magnets were correctly positioned. Using software instead would in theory reduce complexity, and reduce manufacturing costs.

Malfunction 54 error message

Over the course of several weeks, one radiology technician had become very quick at typing commands into the Therac-25 machine. One fateful day, she accidentally entered 'x' for X-Ray rather than 'e' for Electron, so pressed the up key to choose the correct mode. Upon starting the program, the machine shut down, displaying the error "Malfunction 54." Due to the frequency at which other malfunctions occurred, and that "treatment pause" typically indicated a low-priority issue, the technician resumed treatment.

The patient was receiving his 9th treatment, and immediately knew something had gone terribly wrong. He reported hearing a buzzing sound, which was later determined to be the machine delivering radiation at maximum capacity, and feeling as though someone had poured hot coffee over his bath.

After a few days, the patient suffered paralysis due to radiation overexposure, and ultimately died of further complications. The manufacturer believed the root cause was due to an electrical shock, and the machine was put back into service despite an electric company verifying this was not the issue, and that similar incidents had been reported to AECL before.

Reproducing the error

The bug was finally reproduced when the same technician operated the machine on another patient, who also died from radiation overexposure. The hospital physicist was convinced that there was an issue with the machine spreading magnets, and after a lot of trial and error, managed

https://blog.bugsnag.com/bug-day-race-condition-therac-25/

The Worst Computer Bugs in History: Race conditions in Therac-25

to reproduce the issue by performing data entry incredibly quickly.

The dose of delivered radiation was so great that the physicist had to adjust the sensitivity of his detection equipment. The dose was in the range of 10-20,000 rads, which was over 100x the expected dose, and more than enough to kill a grown adult.

Race conditions

The operator's intention was to use the low-power beam, whereas in reality the high-power beam had been used without the spreading magnets in place, delivering a much higher dose than expected. This was due to a race condition within the codebase, which had actually been present in the preceding model, Therac-20, but had been prevented by hardware safety controls.

The software consisted of several routines running concurrently. Both the Data Entry and Keyboard Handler routines shared a single variable, which recorded whether the technician had completed entering commands.

Once the data entry phase was marked complete, the magnet setting phase began. However, if a specific sequence of edits was applied in the Data Entry phase during the 8 second magnet setting phase, the setting was not applied to the machine hardware, due to the value of the completion variable. The UI would then display the wrong mode to the user, who would confirm the potentially lethal treatment.

This bug had actually always been present in the Therac-20 codebase, a fact which was only discovered 2 months after the FDA recall. Hardware safety features on that model meant the error condition had never been

detected, and the code was copied across to the Therac-25 with the cultural assumption that it had been battle-tested.

Byte overflow

An additional concurrency bug caused the last known incident, which was due to overflow in a one-byte shared variable.

Before firing an electron beam, an operator needs to position the machine precisely to target the treatment area. The parameters are then verified by the operator via keystrokes, and "set" pressed, to move the hardware to the correct position.

During verification, the Class3 variable determines whether the hardware is configured correctly. A non-zero value indicates failure, whereas a zero value indicates that everything is setup correctly, and that treatment can proceed.

Because the setup code ran hundreds of times and a byte can only hold 255 possible values, on the 256th attempt of setup, the shared variable would be set to 0.

If at this exact time, the operator was unfortunate enough to hit the "set" button, the program would continue down a codepath which would fire a concentrated X-Ray beam, as the Class3 variable would indicate that the hardware was setup correctly. This did happen, and delivered a lethal dose of radiation to the patient.

Aftermath

Worse still, an internal FDA memo stated that AECL had no formal software specifications or test plan for their device.

Additionally, the software was not evaluated by independent testers, which may have helped combat cultural biases within the organization. A hardware simulator was also used for the majority of development, due to the difficulties of safely testing the actual hardware.

Takeaways

There are several takeaway messages from the whole affair.

Firstly, users will ignore cryptic error messages, particularly if they occur often. "Malfunction 54" does not convey the severity of the machine state, and the average user certainly won't consult the accompanying physical manual to find out what it means.

Secondly, usability can sometimes get in the way of safety. In this case it would have been preferable, although admittedly annoying, to have forced the user to re-enter commands in the case that they made an error, and to review the commands before executing.

A final note is that in safety-critical systems, code should be subject to formal analysis from independent parties from those who developed it. Some level of automated testing at the unit level is also needed, rather than only testing the system as a whole.

Conclusion

Therac-25 is an extreme example of what can go wrong with software systems, and the devastating consequences that bugs can have on regular people.

Although most of us won't work on safety-critical systems, software errors can still have a significant impact on our users. Most of us will have experienced a software failure during an important slideshow presentation, an app on our phones crashing in the middle of an activity, or a data breach that leaks our credentials to the entire internet.

What's the worst thing that could go wrong in your application?

Our series on the Worst Software Bugs in History is in honor of Bug Day 2017. Seventy years ago, Grace Hopper discovered the first computer bug — a moth was stuck between relays in the Harvard Mark II computer she was working on. The notion of bugs was described in other fields previously, but the moth discovery was the first use of the term "debugging" in the field of computers.

Sources:
http://courses.cs.vt.edu/cs3604/lib/Therac_25/Therac_1.html
https://en.wikipedia.org/wiki/Therac-25

Share

https://blog.bugsnag.com/bug-day-race-condition-therac-25/