Nada AMIN
Rinard, TR 12

An Investigation of the Therac-25 Accidents

The Therac-25 was an electronic machine for radiation therapy. This machine caused several accidents, including deaths by massive overdose, because of its malfunctions. This paper describes two problems, one in design and one in testing, and how they contributed to the accidents.

The problem in design was that the Therac-25 did not have independent hardware safety mechanism. Instead, the Therac-25 relied critically on the software. Thus, a bug in the software was more likely to lead to a serious accident – and it did. In fact, the Therac-25's predecessor, the Therac-20, didn't cause any serious accidents by massive overdose, despite sharing many of the flaws of the Therac-25. The Therac-20 avoided those accidents, because it had independent mechanical interlocks for ensuring safety operation. In the Therac-25, these hardware safety mechanisms were not duplicated, probably because their implementation might have appeared to be an unnecessary cost. Given the safety-critical nature of the system, sparing the redundant hardware protection was very risky.

Software testing was also a problem, because it wasn't systematic. The decision to rely on the software almost exclusively might have been understandable, had the software been tested sufficiently thoroughly to justify such a high confidence. Unfortunately, the software clearly wasn't tested methodically. It seemed that the testing consisted mostly of "2,700 hours of *use*" (20). It is unlikely that typical use would capture the subtle cases in which the code fails. Most likely, the thousands of hours would just be repeating a few most common cases – a repetition which is not efficient, nor reassuring. In fact, one bug in the software that triggered at least two tragic accidents would probably have been detected by more thorough testing. The bug was that editing on the screen could possibly go unnoticed while the physical system was set up. This bug was buried in the code as it involved not a single subroutine but the interactions among a few of them. It could have been uncovered by testing whether the system indeed detected changes in input during the physical set up.

In conclusion, the design decision to remove the hardware safety protection was lamentable. Unnecessarily, it forced the software to assume a safety-critical role that could not be justified in face of its problematic testing.