LETTERS TO THE EDITOR

Therac-25 revisited

To the editor:

I am sending this note to thank Nancy G. Leveson and Clark S. Turner for their excellent article on the Therac-25 accidents (*Computer*, July 1993, pp. 18-41).

Preparing this article was a genuine service to the real-time community. For example, I have been using the article in an effort to educate my management about what constitutes adequate qualifications for staff who work on real-time control systems for our chemical plants and refineries.

Norman F. Jerome
Amoco Research Center
Naperville, Illinois

To the editor:

I have a serious question to raise about "An Investigation of the Therac-25 Accidents" by Leveson and Turner:

Where was the doctor?

At several points the authors describe technicians as operating a delicate instrument that exhibited "frequent malfunctions" during routine operation and that issued "malfunction messages" so often that it was standard policy to ignore them. Who was permitted to lay down such a policy?

The combined software and hardware problems are described as having been discovered gradually. Yet why isn't the contribution of medicine recognized? The heroes are said to have been a radiation physicist and a corps of technicians and operating engineers: The impression is of a therapeutic environment in which patients wander into a clinic and pick out a technician just to get a "good dose of radiation treatment." It's not a drug, so why should they need a prescription!

The article suggests that the professional involved, namely the medical doctor prescribing the treatments, did absolutely nothing to discover or prevent the problem. I do not accept this as accurate.

John Michael Williams
Redwood City, California

Authors' reply:

The MD was not involved in the day-to-day operation of the Therac-25; the physicians would write prescriptions for their patients, who would then be scheduled by the hospital for treatment. Obviously, the physicians had to know about the serious results of the accidents, but we could find no information in the boxes of material we received from the FDA and other sources (including two lawsuits for which Nancy Leveson was an expert witness) to indicate any direct involvement of the MDs in finding or resolving the Therac-25 problems. This does not mean that they did nothing, but we could only present what we were able to learn from publicly available documents. We do not mean to imply anything from our omission of information about the doctors' responses except ignorance on our part about them.

To the editor:

I am grateful for the fine analysis of the Therac-25 accidents. At last I'm able to appreciate the vague reports I've heard over the last five years about "software killing patients" under treatment by an "X-ray machine." The agony of the patients injured and killed by the Therac-25 permeates the well-written technical prose.

I'm afraid two other unfortunate impressions reach me: almost criminally bad electromechanical design, and unconscionable clinical usage of the Therac-25.

Safe treatment by the Therac-25 requires the correct mechanical positioning of beam-shaping structures, one for each of the two accelerator modes. If the correct structure is not positioned in the path of the beam, overdosage is likely to occur. I believe it is exceedingly bad design to use an elaborate scheme involving software rather than the simple combination of two cams and two normally open microswitches to interlock the two treatment modes.

Certainly the doctors and other professionals — if not the accelerator technicians — involved in the Therac-25 treatment were remiss in allowing humans to be subjected to an energetic device that issued so many malfunction indications that the operators "became insensitive to machine malfunctions" or that on some days exhibited "40 dose-rate malfunctions." Furthermore, once the suspicion arose that the beam-shaping structures were not properly positioned automatically, it should have been expected of the operators to visually check before beam power was applied.

There isn't space to discuss some of the other design flaws exhibited (consider, for example, an undocumented error message "dose input 2," which means the delivered dose was too high or too low!). And what about the pitiful response of US and Canadian government regulators?

As a programmer, I am critical of any program involving setup followed by actuation that does not *indivisibly* check the setup for consistency, reasonableness, safety, and so forth, immediately before actuation.

The industry's state of the art (and certainly my personal state of the art) in building safe programs is not sufficiently advanced to risk human life. Consequently, I have decided not to write software for high-energy treatment or transportation of humans, or for life support or critical patient monitoring. I would like to correspond with anyone who thinks he or she can responsibly write software when bugs can have lethal consequences.

William E. Drissel
President, CyberScribe Inc.
Grand Prairie, Texas

Authors' reply:

We can only add that the features William Drissel notes, along with other similar problems, are, according to our experience, common in safety-critical software. Perhaps those who write such software should be people exactly like Drissel who appreciate the dangers and are worried about their competence to write the software. Complacency is the most frequent common aspect of major accidents.

To the editor:

I was encouraged to see the issue of software reliability in critical systems brought to the attention of a broad professional audience. However, there is an editorial issue that I feel detracts from the authors' technical credibility.

The editorial point, although minor when compared with the importance of the article's message, is significant for computer professionals dealing with software reliability. The authors use nonstandard terminology in dis-

cussing software faults, failures, and errors. The term "error" is used to refer to faults, the term "fault" is used to refer to failures, and the term "bug" is used in its typically ambiguous way.

IEEE/ANSI Standard 982.2 defines standard terms for discussing software reliability. Researchers working in the software reliability area are undoubtedly aware of this standard. The authors' apparent ignorance of the IEEE standard brings their technical credibility into question. It also reflects poorly on *Computer*'s editorial staff and review process.

Norman Young
Ottawa, Ontario
Canada

Authors' reply:

We are very aware of the standard, but after a great deal of experience in trying to apply these definitions to real software problems, we have found them to be meaningless in practice. This is not the place to go into our objections to the definitions, but we were quite aware that we were not following the IEEE standard. None of the 17 reviewers of the article, most of whom appeared to be software reliability experts, as are the authors, nor any of the hundreds of people who read early versions of the report noted any problem with the terminology or felt that it hindered their understanding.

We have also had enough arguments and communication problems with engineers to recognize the problems that occur when software engineers define terms differently than hardware engineers. To differentiate the type of random, wear-out failures in engineering from design errors of the type that occur in software (and hardware), we use the term "error" to mean a generic design problem with the software. This is also closer to natural language usage, and our goal was communication with a large audience.

We find it indicative of the problems in software engineering and software reliability that lead to accidents such as those described in the article that technical competence is defined as following the definitions of terms in IEEE standards.

To the editor:

"An Investigation of the Therac-25 Accidents" is an excellent example of technical reporting conducted under difficult circumstances due to the legal sensitivities. However, I must comment on the authors' viewpoint for fault analysis and recommendations. The major weakness of the Therac-25 system was not in hardware or software, but in design. The overwhelming, dominant flaw was that the system was not designed to measure radiation dosage throughout the range it was capable of producing. Leveson and Turner write: "The Therac-25 software 'lied' to the operators, and the machine itself could not detect that a massive overdose had occurred. The Therac-25 ion chambers could not handle the high density of ionization from the unscanned electron beam at high beam current; they thus became saturated and gave an indication of low dosage. Engineers need to design for the worst case."

One cannot blame the software for lying if it is accurately reporting the output of the sensors. And the sensors apparently always worked exactly as designed.

If the operators had been informed that they were administering fatal doses of radiation to patients, I very much doubt if they would have repeated the treatments, no matter how simple the command or how mundane the alert. I also believe that the AECL (Atomic Energy of Canada Limited) would have quickly recognized and isolated the software faults if they had been presented with hard physical measurement of the massive dose rates, rather than muddling for almost a year in ambiguous investigations and claiming that overdoses were impossible and no other accidents had occurred previously.

I agree wholeheartedly that "there is always one more bug," especially in systems that deal with concurrent, unsynchronized processes. One cannot disagree that relevant training and experience will reduce the number of programming errors, but they will never be eliminated. Redundant safety systems reduce the number of accidents, but they become prohibitively expensive and frustrate normal operation with false alarms as systems become more complex. The only path to arbitrarily complex reliable systems is through unambiguous detection of errors as they occur. The amount of monitoring information collected from a system will always have to be balanced against the amount of effort required to isolate the fault when an error occurs. However, the absolute minimum monitoring requirement is that it should detect any error that could cause fatal injury. The Therac-25 design failed to meet this minimum requirement.

Thomas D. Gamble
Annandale, Virginia

Authors' reply:

We have to do a lot more than just detect that we have already made a mistake, especially when that mistake is not correctable, such as having given a fatal dose of radiation to a patient. We agree that error checking is important, but it is not the complete answer to preventing accidents: Some errors are not detectable, some are detectable but no recovery path exists at that point, and some are not detectable until after the damage has already been done. There is always a limit (in terms of memory, execution time, etc.,) to how much checking is practical: Checking everything is prohibitively expensive. Accidents usually involve the occurrence of events that everyone thought were impossible before they occur (or they would have taken preventative measures such as putting in checks and interlocks). We said in our article, as the letter writer notes, that they should have designed for the worst case, but a simple message that the worst case has already occurred is not the solution to preventing accidents.

Furthermore, AECL was very much aware that an overdose had occurred after the Hamilton accident (within six weeks of the first overdose), and they even knew that it was caused by an unscanned beam. Yet this did not allow them to "quickly recognize and isolate the software faults" as the letter writer suggests. Knowing that something bad has occurred in a system does not necessarily provide any evidence that software was involved. Hindsight is always perfect, but there are no simple answers to complex problems.

Nancy G. Leveson
University of Washington

Clark S. Turner
University of California, Irvine

One voice or many?

To the editor:

In his March 1993 editorial, Editor-in-Chief Ted Lewis stimulates discussion among IEEE members on the roles that government policies can play in support of national competitiveness.

As chair of the IEEE's United States Activities Board, I preside over a volunteer structure of more than 30 committees composed of approximately 750 US IEEE members. Work

ing with the IEEE-USA staff in Washington, members develop and deliver recommendations to Congress and the executive branch on a broad array of issues, including industrial policy and US competitiveness.

IEEE-USA promotes the professional careers and technology policy interests of US members. The unit presents its views through position statements, testimonies, workshops, and symposia. Facilitating defense conversion and technology commercialization is a current challenge. For a copy of our latest Legislative Agenda and/or a listing of our positions, please direct inquiries to Chris Brantley at IEEE-USA in Washington, phone (202) 785-0017, e-mail c.brantley@ieee.org.

I fully appreciate the difficulties a multinational society faces in addressing issues that may interest only a portion of its membership, or on which there may be conflicting views. However, with 37 IEEE technical societies, I would also question the desirability of having the institute speak to US policymakers in 37 different voices.

The simple fact is that US members constitute approximately 75 percent of the total IEEE membership. We need an effective mechanism to voice US members' perspectives and concerns on government policies that affect their interests.

We must also have input from our technical societies, especially in providing names of knowledgeable and interested volunteers to serve as liaison members to the IEEE-USA Technology Policy Council committees. We must ensure that technical society perspectives are an integral part of our policy development activities from the beginning of the process. We encourage and welcome the societies' valuable contributions.

Charles X. Alexander
Vice president, professional activities
IEEE-USA

Author's reply:

I was vaguely aware of this group's activities, which is one reason I raised the question. If you believe in what this group is doing, I encourage you to volunteer your expertise. Furthermore, I would like to invite Charles to periodically update this editor on contributions made by the USAB. If the notice is something our dues-paying members might be interested in, we will print it.

Ted Lewis, editor-in-chief

ARTICLE SUMMARIES

Interactive Scientific Visualization of Fluid Flow, pp. 13-25

Paul R. Woodward

The demands of interactive response require simplifications in the algorithms used to render data, particularly in the case of three-dimensional fluid-flow simulations. This article gives examples of scientific visualization techniques used for the interactive exploration of very large data sets from supercomputer simulations of fluid flow.

Interactive rendering of images from simulations on grids of 2 million or more computational zones are required to drive today's high-end graphics workstations to their limits with 2D data. The author presents one such image and discusses interactive steering of 2D flow simulations, a phenomenon now possible with grids of half a million computational zones.

The author uses a simulation of compressible turbulence on a grid of 134 million computational zones to set the scale for discussing interactive 3D visualization techniques. Each of the 170 snapshots of this flow requires a 1.3-gigabyte data file. Perspective volume rendering techniques implemented on massively parallel computers make possible interactive rendering of images from such simulations. Interactive selection of smaller subvolumes for rendering allows high-end workstations to be used to explore such data sets.

The article concludes by discussing a concept for a gigapixel-per-second Video Wall, or Gigawall, which could be built with present technology to meet the demands of interactive visualization of the data sets that will be produced by the next generation of supercomputers.

Parallel Finite-Element Computation of 3D Flows, pp. 27-36

Tayfun Tezduyar, Shahrouz Aliabadi, Marek Behr, Andrew Johnson, and Sanjay Mittal

The authors describe their work on the massively parallel finite-element computation of compressible and incompressible flows with the Connection Machine CM-200 and the CM-5. Their computations are based on implicit methods, and their parallel implementations are based on the assumption that the mesh is unstructured.

Computations of flow problems involving moving boundaries and interfaces are achieved by using the Deformable-Spatial-Domain/Stabilized-Space-Time method. In this method, with special mesh update schemes, the frequency of remeshing is minimized to reduce the projection errors involved in remeshing and also to make parallelizing the computations easier. This method and its implementation on massively parallel supercomputers provide a new capability for solving a large class of practical problems involving free surfaces, two-liquid interfaces, and fluid-structure interactions.

Current three-dimensional incompressible-flow computations are being carried out at sustained speeds of 1.2-2.1 gigaflops on the CM-200 (with 1,024 processing nodes) and 3.6-4.6 Gflops on the CM-5 (with 512 processing nodes). The 3D matrix-free implicit compressible-flow computations are carried out at sustained speeds of 9.5-10.4 Gflops on the CM-5. This parallel performance signifies a new level of computational capability for the finite-element solution of 3D flow problems.

The 3D problems solved include sloshing in a liquid-filled container subjected to vertical vibrations, incompressible flow between two concentric cylinders, supersonic flow past a delta wing, and supersonic flow past a toy missile.