New Media

D. John Doyle MD PhD FRCPC

Computer programming errors in medicine

A recent issue of the New England Journal of Medicine¹ contained an unusual retraction. This time the concern was not the discovery of a fraudulent research report, a sickening problem that occasionally strikes that most prestigious journal but, rather, the retraction arose from a computer programming error. Specifically, in a study of suicide rates following natural disasters, the data analysis software had a previously undetected mistake that resulted in the software erroneously counting some deaths twice.

The authors acted ethically in reporting this error and retracting their paper, despite the embarrassment and career implications inevitably involved. How many others would have merely kept quiet and not issued a retraction, especially knowing that failure to weed out this misinformation would not likely result in any patient harm (as compared to, say, an error in a dose-finding study)? This problem also raises the issue of the role of senior clinical investigators, most with very limited programming skills, in supervising the efforts of the programmers that work on their research team. Such errors have long been the concern of computer scientists and software engineers, and a very enlightening moderated electronic forum, known as The Risks Digest² exists for the discussion of these issues. Because this forum is moderated (unlike, for instance, the GASNet Anesthesiology Forum³ and most other clinical discussion groups), the discussions are edited to ensure that they are well-written and relevant. As a result, the postings are sufficiently limited in quantity that they are easily browsed or searched. (Engineers talk about such a forum as having an excellent "signal-to-noise ratio".)

While the NEJM programming error is unlikely to put anyone's life at risk, other computer programming errors have been less benign. For instance, one computer error known as a "race condition" resulted in individuals being over-radiated (to death in two cases) in Texas and Georgia while receiving cancer therapy using a Canadian-designed AECL Therac-25 radiation treatment system.⁴ The malfunction was caused by a software error in the computer program controlling the machine. The problem was a subtle error that no one had detected during the extensive testing the machine had undergone before being introduced into clinical use. The error surfaced only when an operator happened to use a specific, unusual combination of keystrokes to instruct the machine about the radiation parameters to be used.

Specifically, an investigation revealed that if an extremely fast-typing operator inadvertently selected the X-ray mode instead of the electron beam mode, and then used an editing key to correct the command to select the electron mode instead, it was possible for the computer to lag behind the orders. The result was that the device appeared to have made the correction but in fact still had incorrect settings.

Although the technical details of the failure remain secret as a result of a legal settlement, experts have come up with the following account as the most likely accident scenario. A modern radiation therapy machine is based on a linear accelerator that produces a high-energy electron beam. One may direct the electrons directly into the patient, or, to produce X-rays, one places a heavy metal target in the electron beam, so that when the electron beam hits the target, X-rays come out from the other end. The target is moved in

Address correspondence to: D. John Doyle MD PhD FRCPC; New Media Editor, Canadian Journal of Anesthesia, Department of Anesthesia, The Toronto Hospital, 200 Elizabeth Street, Toronto, Ontario Canada M5G 2C4 Voice pager: (416) 375-0565; Fax: (416) 423-0452; E-mail: djdoyle@inforamp.net

and out of the beam automatically under software control, depending on whether an electron beam or an X-ray beam is selected to treat the patient. Also, the current in the electron beam is programmed to be much greater in X-ray mode because of energy losses that result when the target is used in making X-rays. Because of the software error, the computer “thought” it was in X-ray mode rather than in electron mode, resulting in a radiation overdose.

How could this be allowed to happen? Experts speculate that the software developers might not have considered it necessary to guard against this failure mode or might not even have imagined it, since radiation therapy designers have traditionally used electro-mechanical interlocks to ensure safety in this setting. Also, systems analysts reviewing the case noted that the unit should have been programmed to discard “unreasonable” readings - as the injurious setting presumably would have been. Finally, there should have been no way for the computer’s verifications on the video screen to become unsynchronized from the keyboard commands.

These concerns raise the issue of whether a certification process for the development and testing of medical software should be in place. Interested parties include all medical software developers, medical equipment manufacturers, the American Food and Drug Administration (FDA), Canada’s Health Protection Branch and, of course, a variety of standards agencies (ISO, CSA, UL, CEN, ASTM, IEEE, ANSI). Most efforts emphasize good software development practices and special safety measures appropriate to the clinical setting.

One particularly active player in this field is the FDA, who provides a lot of useful resources at their web site⁵ (start with the medical devices section). The FDA defines software as a “set of instructions that enables a computing machine to control, monitor or otherwise interact with a medical device.” Their proposed regulations for medical software developers would require a software developer to show that the algorithm, or mathematical technique, used in the computer program has been correctly implemented in software. The FDA also requires assurance, through a “risk analysis” that any software failure could not injure a patient. How that assurance can be provided in general is still unclear, since any risk analysis is necessarily linked to the specifics of the application. For instance, clinicians will instantly recognize that software errors in controlling a sodium nitroprusside infusion are far more likely to result in patient injury than, say, a real-time phonocardiographic monitoring system designed to detect the new onset of cardiac mur-

murs in CCU patients (for example, with suspected ischemia of the mitral valve musculature). Techniques for evaluating software safety are relatively new, drawn in part from the aviation and nuclear industries as well as from academia. Who does the checking, how much evidence is enough and whether the FDA or other authority can perform an independent check of the software are important regulatory and cost issues. Furthermore, software developers may be wary of submitting complete listings of their computer programs because of concerns that competitors might get a look at this “source code” via a request under the American Freedom of Information Act.

One advocated approach to this challenge is to begin with the idea that not all computer errors are equally serious, as with the example given above where drug controller systems are seen to be riskier than, say, a simple monitor. Thus, risk analysis techniques would require more exhaustive safeguards in software developed for risky applications compared with applications that are inherently safer. Software developers must also continue to improve the methods they use for documenting, writing and testing and maintaining medical computer programs. The need for software development methods that insures that programs are written in a consistent, easily understood and reliable way must be seen to be paramount. Too often, programmers include a description of what each part of their program does (the “documentation”) only as an afterthought, rather than starting their software writing from specifications provided in a carefully developed design document.

As with buggy commercial software, in the rush to the marketplace, when delays can put a company at a competitive disadvantage, software testing may lose out. This can happen even in the medical environment. Delays in releasing a software package to allow additional testing must be balanced against the possibility of failing to detect any errors. When financial losses or injury result from software defects, the situation can lead to lawsuits. Software is clearly a product, and if it is defective and injures a consumer or a patient, then the manufacturer may be liable. To many medical-device producers, the threat of litigation may be even more effective than government regulations for assuring the quality of medical software products.

As we enter the new millennium, many of us are curious about the possible clinical implications of any Y2K bugs (Y2000 problems) that may (or may not) exist in anesthesia machines, infusion pumps or even implanted defibrillators⁶ (an internal clock/calendar being present for the telemetry reporting of arrhythmias and discharge events). As everyone knows, the

Y2K problem arises from a long-standing and often-used programming practice whereby affected programs use only two digits rather than four to represent years. This may lead to a variety of computer malfunctions and data errors related to crossing from 1999 (99) to 2000 (00). In this setting, software may interpret 00 as 1900 or another incorrect date, or may even "crash", resulting in a loss of service situation. Clinicians may be seriously affected by this problem as it relates to the operation of clinical equipment. As well, business functions such as scheduling, billing and purchasing; the reliability of infrastructure such as power and telecommunications; the availability of supplies; and many other issues, are involved in this nasty problem. In many cases upgrading to the latest software release easily solves the problem. Already, lawsuits have been initiated by individuals who felt that it was unfair to have to pay for fixes to Y2K defects.

References

- 1 Krug EG, Kresnow M, Peddicord JP, et al. Retraction of "Krug EG, Kresnow M, Peddicord JP, et al. Suicide after natural disasters. N Engl J Med 1998; 338: 373-8". N Engl J Med 1999; 340: 148-9.
- 2 Risks Digest. Forum on risks to the public in computers and related systems. <http://catless.ncl.ac.uk/Risks>.
- 3 <http://groucho.med.yale.edu/discussion/anesthesiology/>
- 4 Peterson I. A digital matter of life and death. Science News, 12 March 1988.
- 5 <http://www.fda.gov>
- 6 Anonymous. Year 2000 compliance issues. Health Devices 1999; 28: 104-12.

Erratum

Choi J-H, Rooke GA, Wu S-C, Bishop MJ. Reduction in postintubation respiratory resistance by isoflurane and albuterol. Can J Anaesth 1997; 44: 717-22.

Please note the equation on page 718 (July issue) is incorrect. It should read:

$$P_i = V/C + R_f \text{ and}$$

$$P_e = V/C - R_{fe}$$