Therac-25: Will history repeat itself?

Ash Tyndall

CITS5502 Software Processes
2014

Contents

CHAPTER 1

Introduction

In the early 1970s, two companies, Atomic Energy Canada Limited (AECL) and Compagnie General Radiographique (CGR), collaborated to build updated models of their core radiotheraputic offerings; Medical Linear Accelerators (LINACs). LINACs are an extension of the basic concept of a linear particle accelerator, repackaged for medical applications. They are are designed to create a beam of either electron or x-rays which can be focused on a very specific section of a patients body. These beams can be used to kill cancerous growths without severely damaging surrounding tissue.

Together, AECL and CGR develop two LINACs; the Therac-6 and the Therac-20, both of which were based on previous CGR work, but with mechanical control substituted with control via computer terminal. These machines are designed with accelerators that could produce 6 MeV x-rays and 20 MeV x-rays/electrons respectively. Still in partnership in the mid-1970s, AECL and CGR develop a new Therac that uses a new kind of linear accelerator, a "double-pass" system, which reduces the space requirements and allows cheaper parts to be used. AECL and CGR part ways soon thereafter, citing competitive pressures.

AECL proceeded with the development of this new "double-pass" system, the Therac-25 (so named for its 25 MeV x-ray and electron beam), continuing to develop the software-based control system, and passing the necessary regulatory requirements. The system is announced for public sale in 1983. One of the cutting-edge features of the machine is the removal of hardware-based interlocks and safeguards on the device, AECL instead opting to use a wholly software-based approach to ensure that the appropriate components are rotated in front of the raw electron beam to reduce the dangerous radiation to therapeutic levels.

In 1985, the first reported case of Therac-25 software safeguard failure occurred. Katherine Yarbrough, a breast cancer patient, receives an estimated 15,000–20,000 rads instead of the normal 200 rad range. She suffers severe radiation burns and shoulder and arm paralysis. A month later at a different facility, an unidentified female patient receives four separate overdoses totalling

13,000–17,000 rads within the space of several minutes, due to a combination of safeguard failure and poorly explained error messages. This patient dies of radiation induced cancer some months later.

It was initially unclear to those who operated the Therac-25 that software errors were the cause of these overdoses. Due to the nature of radiation overdoses, the most serious of symptoms appear days if not weeks later, causing the resulting deaths and disablements to be attributed to other factors. However, over time, it became clear to different system operators that something was seriously wrong with the Therac-25 , sparking an eventual forced FDA recall of the product with a total of six overdoses and two deaths.

Therac-25 was responsible for the first known deaths in radiotherapy, a profession that began some 35 years prior, and the confluence of death and computing caused an uproar at the time. Since then, the flames of anger have died down, and there is an opportunity to review the incident objectively. This report will examine the Therac-25 case in detail, trying to determine the answers to several important questions:

1. How does one design safe software, what does it involve, what are the pitfalls and how do we avoid them? (Chapter 3 on page 5)
2. What were the flaws of Therac-25 and how did failures on the part of AECL and CGR contribute to the creation of these flaws? (Chapter 4 on page 6)
3. How did standards and regulatory bodies respond to Therac-25 through new guidelines and processes for creation of "safe" medical device software? (Chapter 5 on page 7)
4. Have those guidelines and processes resulted in the creation of safer medical device software? (Chapter **??** on page ??)
5. Are there still areas in the medical landscape where regulation is insufficient? (Chapter 7 on page 9)
6. Will Therac-25 happen again? (Chapter 8 on page 10)

CHAPTER 2

Literature Review

Therac-25 was one of the first widely reported software-related disasters, and as a result, a variety of academic work has been performed since the disaster investigating the specifics of the failure of Therac-25 , as well as the development of safety critical medical software generally.

Primary work in the area of safety critical software development and system design as been done by Nancy Leveson. Her two books *Safeware* [6] and the more recent *Engineering a Safer World* [7] are highly influential works in the field. They discuss from a variety of disciplines the causes of safety-critical system failure, from poor software testing processes to a failure of system design to adequately inform the user of the system's state. Of particular relevance to our Therac-25 questions is *Safeware* chapter 6, "The Role of Humans in Automated Systems", which describes several models of human-computer interaction and the necessary design principles to properly enable them. Additionally, *Engineering a Safer World* chapter 9, "Safety-Guided Design", which discusses several of the Therac-25 design flaws.

Leveson and Turner have also contributed the primary academic report on Therac-25 [8] which discusses the history of the Therac brand, the history of the companies involved, as well as the timeline of the disasters that ensued.

Various work has been done into the area of safe critical software from both the perspective of cause and prevention. Dunn [3] provides us with a helpful definition of safety in terms of "mishap risk", as well as examples of mishap causes.

In terms of cause, in *Failure in Safety-Critical Systems* chapter 3 [4] Johnson discusses in detail the sources of failure, touching upon a broad variety of failures including Regulatory, Managerial, Hardware, Software, Human and Team based failures. Besnard and Baxter et al. [1] discuss two models of system failure, Reason's swiss cheese model and Randell's fault-error-failure model, both of which can be used to analyse the Therac-25 disaster.

In terms of prevention, Nolan [10] proposes several strategies to be considered when designing “safe systems of care”, as well as methods to reduce “adverse events” which may compromise patient health. Obradovich and Woods [11] perform an investigation of poor Human-Computer Interface design in a medical device, describing how the device is flawed and how both the user and medical supervisor can change their processes to cope with this. Lin, Vicente and Doyle [9] propose a new interface for a specific medical appliance that applies human factors engineering, a key part of the discussed prevention of user error, and provides data demonstrating the effectiveness of such an approach from error minimisation and efficiency perspectives.

Several works discuss the introduction of various international standards and regulations post-Therac-25 . Rakitin [12] provides an overview of foundational standards in the risk management space of medical device software. Brown [2] provides an overview of specifically the IEC 61508 “Design of electrical / electronic / programmable electronic safety-related systems” standard. Jordan [5] provides an overview of specifically the IEC 62304 “Medical Device Software – Software Lifecycle Processes” standard.

To allow us to determine if regulation of medical device software has improved the situation, several works which analyse relevant data will be examined. Wallace and Kuhn produced two papers [14, 15] analysing a subset of U.S. Food and Drug Administration (FDA) data relating to “adverse events” and provided statistics on the types of software errors and the medical domain of the devices, as well as information on how to prevent and detect these types of errors. The FDA Manufacturer and User Facility Device Experience Database (MAUDE) dataset [13] is also examined directly by the author to attempt to derive conclusions regarding the proportion of medical device software errors in the broader MAUDE database. Finally, *Failure in Safety-Critical Systems* chapter 5 [4] discusses the under-reporting of incidents, as well as reporting bias.

CHAPTER 3

Safe software design: What does it involve?

CHAPTER 4

Flaws and failures: How and why?

CHAPTER 5

Regulatory response: New standards

CHAPTER 6

Data analysis: Are we safer?

CHAPTER 7

Regulatory gaps: What's next?

CHAPTER 8

Conclusion: Will history repeat?

Bibliography

[1] BESNARD, D., BAXTER, G., ET AL. Human compensations for undependable systems.

[2] BROWN, S. Overview of IEC 61508. design of electrical / electronic / programmable electronic safety-related systems. *Computing & Control Engineering Journal 11*, 1 (2000), 6–12.

[3] DUNN, W. R. Designing safety-critical computer systems. *Computer 36*, 11 (2003), 40–46.

[4] JOHNSON, C. *Failure in Safety-Critical Systems: A Handbook of Accident and Incident Reporting*. University of Glasgow Press, 2003.

[5] JORDAN, P. Standard IEC 62304-medical device software-software lifecycle processes. In *Software for Medical Devices, 2006. The Institution of Engineering and Technology Seminar on* (2006), IET, pp. 41–47.

[6] LEVESON, N. *SafeWare: System Safety and Computers*. Computer Science and Electrical Engineering Series. Addison-Wesley, 1995.

[7] LEVESON, N. *Engineering a Safer World: Systems Thinking Applied to Safety*. Engineering systems. MIT Press, 2011.

[8] LEVESON, N. G., AND TURNER, C. S. An investigation of the Therac-25 accidents. *Computer 26*, 7 (1993), 18–41.

[9] LIN, L., VICENTE, K. J., AND DOYLE, D. J. Patient safety, potential adverse drug events, and medical device design: a human factors engineering approach. *Journal of biomedical informatics 34*, 4 (2001), 274–284.

[10] NOLAN, T. W. System changes to improve patient safety. *BMJ: British Medical Journal 320*, 7237 (2000), 771.

[11] OBRADOVICH, J. H., AND WOODS, D. D. Users as designers: How people cope with poor HCI design in computer-based medical devices. *Human Factors: The Journal of the Human Factors and Ergonomics Society 38*, 4 (1996), 574–592.

[12] Rakitin, R. Coping with defective software in medical devices. *Computer 39*, 4 (2006), 40–45.

[13] U.S. Food and Drug Administration. Manufacturer and User Facility Device Experience Database (MAUDE), 2014. Accessed: 2014-09-20, URL: `http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/ReportingAdverseEvents/ucm127891.htm`.

[14] Wallace, D. R., and Kuhn, D. R. Lessons from 342 medical device failures. In *High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on* (1999), IEEE, pp. 123–131.

[15] Wallace, D. R., and Kuhn, D. R. Failure modes in medical device software: an analysis of 15 years of recall data. *International Journal of Reliability, Quality and Safety Engineering 8*, 04 (2001), 351–371.