


The Therac accident

Wednesday, 23 February
2005

1



The Therac-25 Case

- ✍ Computerized radiation therapy machine (safety-critical system) used to zap tumors with high energy beams. It was developed by AECL (Atomic Energy of Canada Limited)
- ✍ Between June 1985 and January 1987 six known accidents involved massive overdoses

Wednesday, 23 February
2005

2

Therac-25 Background

- ⌘ There were two previous versions of Therac machines:
 - ⌘ Therac-6, 6 MeV accelerator capable of producing x-rays only
 - ⌘ Therac-20, 20 MeV dual mode (electrons and x-rays) accelerator
- ⌘ Changes in Therac 25
 - ⌘ Uses a new 'double pass' technique to accelerate electrons (more energy in less space)
 - ⌘ Extensive computer control of the machine (software now coupled to the rest of the system and responsible for safety checks)



Hardware safety interlocks removed

Wednesday, 23 February
2005

3

How Therac-25 Works...

- ⌘ Therac-25 modes:
 - ⌘ Low energy electrons beam through a scan magnet
 - ⌘ X-ray (25 MeV): a foil and a flattener placed in the path of the electron beam are required
 - ⌘ Field light: allowed a standard light beam to shine in the path of treatment to help the operator in setting up the machine (just for setup)

Wednesday, 23 February
2005

4

How Therac-25 Works...

- ✍ In order to get each of these three assemblies in the right place at the right time, the therac-25 designer placed them on a **turntable**

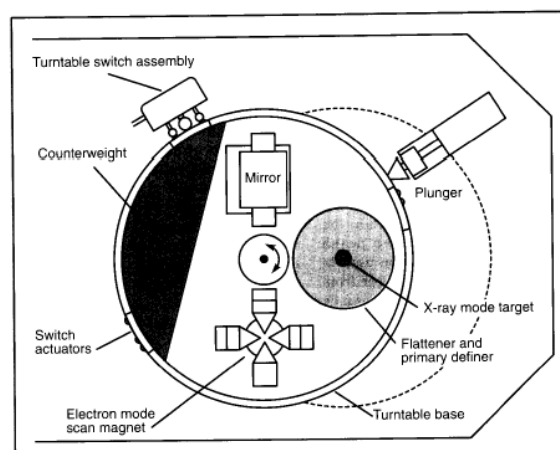


Crucial piece of the Therac-25 machine, since incorrect matching of the turntable and the mode of operation could produce potentially fatal levels of radiation

Wednesday, 23 February 2005

5

The turntable



Wednesday, 23 February 2005

6

How Therac-25 Works...

- ✍ The SW is responsible for monitoring machine status
- ✍ It accepts input about treatment desired, sets machine up for treatment
- ✍ Turns beam on, activated by operator command
- ✍ Turns beam off when treatment is completed, or when operator commands it or when a malfunction is detected

Wednesday, 23 February
2005

7

How Therac-25 Works...

- ✍ The unit has an interlock system designed to remove power to unit when there is a HW malfunction
- ✍ The computer monitors interlock system and provides diagnostic messages
- ✍ Depending on fault, the computer either prevents a treatment from starting or if treatment is in progress, creates a pause or suspension of treatment

Wednesday, 23 February
2005

8

What happened

Wednesday, 23 February
2005

9

Therac-25 Bugs

- ✗ Race conditions
 - ✗ Several different race condition bugs
- ✗ Overflow error
 - ✗ The turntable position was not checked every 256th time a control variable (class3) was incremented
- ✗ Wrong information on the console
- ✗ Non-descriptive error messages
 - ✗ “Malfunction 54”
 - ✗ “H-tilt”
- ✗ User-override-able error modes

Wednesday, 23 February
2005

10

Therac-25 Software Development and Design

- ✍ SW for Therac-25 developed by a single person using PDP11 ASSEMBLY language
- ✍ Developed over several years
- ✍ SW 'evolved' from Therac-6, also taking some Therac-20 subroutines
- ✍ Very little SW documentation produced during development
- ✍ AECL also had an apparent lack of documentation on SW specifications

Wednesday, 23 February
2005

11

Therac-25 SW Testing

- ✍ Manufacturer said the HW and SW were 'tested and exercised separately or together over many years'
- ✍ In deposition, the quality assurance manager explained, testing was done in two parts:
 - ✍ On a simulator
 - ✍ On system
- ✍ Reports indicate that unit and SW testing was minimal

Wednesday, 23 February
2005

12

Therac-25 SW Testing

- ✗ Most testing efforts directed to integrated system test
- ✗ Same QA (Quality Assurance) manager at a therac-25 users meeting stated the SW was tested for 2,700 hours

Lack of a documented test plan!

Wednesday, 23 February
2005

13

The Safety Analysis Report

- ✗ Programming errors have been reduced by extensive testing on a HW simulator and under field conditions on teletherapy units. Any residual SW error is not included in the analysis
- ✗ Program SW does not degrade due to wear, fatigue, or reproduction process
- ✗ Computer execution errors are caused by faulty HW components and by 'soft' (random) errors induced by alpha particles and electromagnetic noise
- ✗ The **fault tree** does *not* include computer failure but only hardware failures

Wednesday, 23 February
2005

14

Contributing Factors

- ✍ Management inadequacies and lack of procedures for following through on all reported accidents
- ✍ Overconfidence in the software and removal of hardware interlocks
- ✍ Presumably less-than-acceptable software engineering practices
- ✍ Unrealistic risk assessment along with overconfidence in the result of these assessment
- ✍ Software reuse

Wednesday, 23 February
2005

15

Lesson Learned

- ✍ An accident is not determined by a single contributing factor
- ✍ Fixing a particular error does not prevent future accidents, there is always another software bug: never put too much confidence in the software
- ✍ System should not be designed so that a single software error (or software engineering error) can be catastrophic
- ✍ Documentation should not be an afterthought

Wednesday, 23 February
2005

16

Lesson Learned

- ✍ Design should be kept simple
- ✍ Ways to get information about errors should be designed into the software from the beginning
- ✍ The software should be subjected to extensive testing and formal analysis at the module and software level: system testing alone is not adequate
- ✍ Special safety analysis and design procedures must be incorporated into safety-critical systems
- ✍ Reusing software does not guarantee safety in the new system

Wednesday, 23 February
2005

17

Conclusion

- ✍ We must approach the problem of accidents in complex systems from a system-engineering point of view and consider all possible contributing factors
- ✍ Any engineer is not automatically qualified to be a software engineer

Wednesday, 23 February
2005

18