Therac-25 Medical Device

Adapted from: Leveson, N, "Safeware: System Safety and Computers", Addison-Wesley, 1995.
Leveson, N., Turner, C., S., An investigation of the Therac 25 Incidents, IEEE Computer, July 1993

24 November 2009 1

Background

- **The medical device destroys tumors in shallow tissue with accelerated <u>electrons</u>, and deeper tissue are destroyed by <u>X-ray</u> photons**
- **In early 1970s Energy Atomic of Canada (EACL) and a French company CGR built together linear accelerators**
  - ➔ Therac-6 , 6 MeV (6 million electron-volt)
  - ➔ Therac-20, 20 MeV

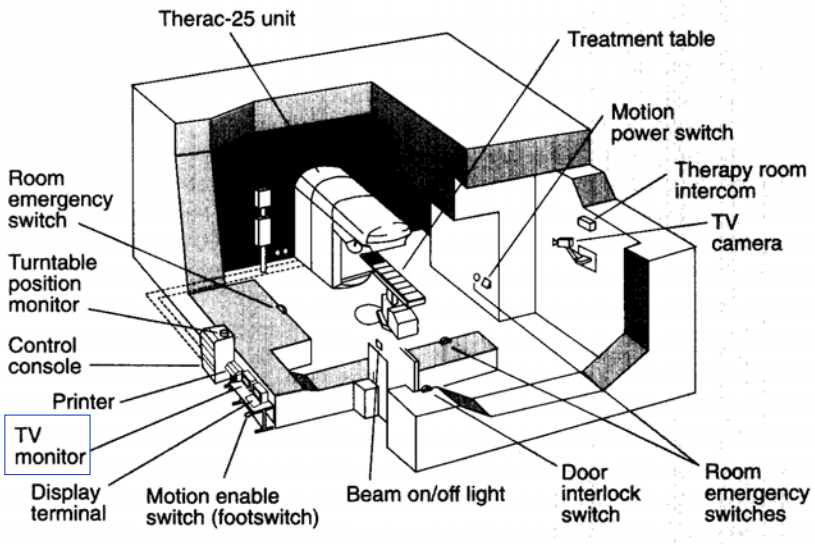
24 November 2009 2

Background

- **Therac-6 software was developed by the French company**
- **Software functionality was limited**
  - ➔ Machines were capable of operating without software
  - ➔ Added convenience to hardware
  - ➔ Interlocks were retained in machines
    - Interlock is device used to help prevent a machine from harming its operator (in this case a patient) or damaging itself
- **Therac-6 software was <u>reused</u> in Therac-20**
- **Business relationships faltered after Therac-20 effort**
- **EACL designed the new model, the Therac-25**

24 November 2009 3

Typical Therac-25 Facility



24 November 2009

Leveson, IEEE Computer, July 1993 4

Therac-25 Features

- **Developed in mid-1970s using a PDP-11 computer**
- **Can deliver 25 MeV photons or electrons at various energy levels**
- **Software has more responsibilities**
  - Safety
  - AECL decided not to duplicate existing hardware safety mechanisms and interlocks
- **Some of the old Therac-6 software was reused**
- **Quality Assurance (QA) manager was not aware that old Therac-20 software was reused**
  - This was only discovered after an accident with the Therac-25

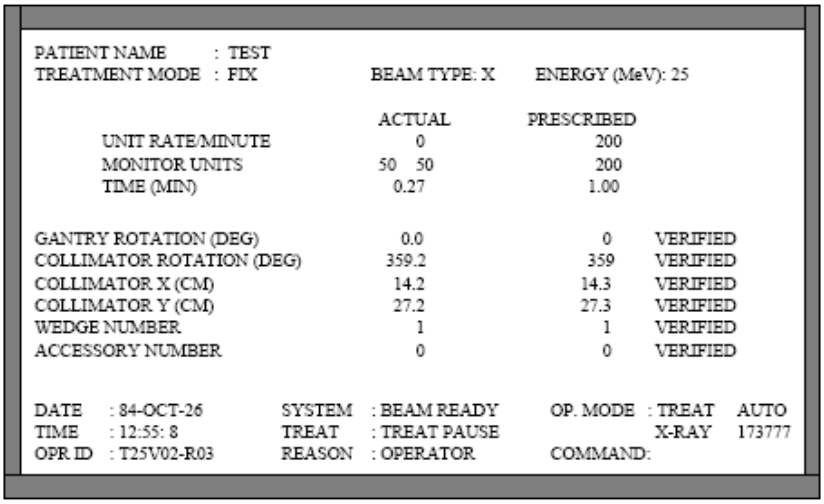
24 November 2009 5

Therac-25 Features

- **A Turntable Rotates Equipment to Produce Therapeutic Modes**
  - Electron and photon modes
  - Another mode is used to position correctly the patient
  - Traditionally, electromechanical interlocks were used to ensure safety
- **Operator Interface**
  - Terminal is used to enter patient identification and treatment prescription
  - Error messages were cryptic: e.g. numbers 1 to 64
    - Operator manual does not describe malfunction codes *
  - Software does not contain a safety feature to prevent excessive radiation being delivered to patients

24 November 2009 6

Operator Screen Interface



24 November 2009 7

Therac-25 Events

- **Safety Analysis was Performed in 1983**
  - Apparently excluded the software
  - Probability of computer selecting wrong energy
    - 10 ^11 probability
    - no justification for this number
- **Events**
  - 11 machines were installed: 5 in US and 6 in Canada
  - 6 accidents occurred between 1985-87
  - Machines were recalled to make extensive changes
  - It was found that old Therac-20 software errors caused accidents in Therac-25.
    - Software errors were not discovered in Therac-20 because of hardware mechanisms

24 November 2009 8

Therac-25 Events

- **Kennestone Regional Oncology Center, June 1985**
  - Patient received one or two doses of 15,000 to 20,000 rads
    - 1 rad = .01 Joule/Kg
  - Typical single therapeutic dose are in the 200 rad range
  - 500 rads is the accepted figure for whole body radiation that will cause death in 50% of the cases
- **Ontario Cancer Foundation, July 1985**
  - Operator pushed the button
    - Operator console displayed " **NO DOSE**"
  - Operator went along with second attempt
  - Operator repeated this process four times
  - Patient complained of burning sensation
  - Patient had received between 13,000-17,000 rads

24 November 2009 **Video clip** 9

Therac-25 Events

- **US Food and Drug Administration (FDA) report**
  - "Material submitted by the manufacturer has not been in sufficient detail and clarity to ensure an adequate software quality assurance program currently exists."
  - "In addition an analysis has not been provided to demonstrate the corrected software does not adversely affect other software functions
- **EACL has not planned on any quality assurance testing to ensure exact copying of software**
- **EACL QA Manager**
  - Software and hardware changes to be retrofitted following the Tyler accident nine months earlier, but which had not yet been installed, would have prevented the Yakima accidents

24 November 2009 10

Therac-25 Events

- **EACL QA Manager**
  - Tests had been done on changes, but tests were not documented and independent evaluation of the software "might not be possible"
  - Outside experts had reviewed the software, but he could not provide their names.
  - An outside consultant performed a review (i.e. an inspection), no information is provided in the final safety report about methodology or tool used
- **FDA**
  - The test data presented to show that the software changes to handle the edit problems are appropriate proved the exact opposite results.

Findings

24 November 2009 11

Lessons Learned

- **Overconfidence in Software**
  - Feeling that software will not or cannot fail
  - Safety analysis did not include software
- **Confusing Reliability with Safety**
  - This software was highly reliable, it worked tens of thousands of times before overdosing anyone
  - AECL assumed that their software was safe because it was reliable
- **Lack of Defensive Design**
  - Software did not contain self-checks, error detection or error handling features
  - Patient reactions were the only real indication of the seriousness of the problems

24 November 2009 12

Lessons Learned

- **Engineers Need to Design for the Worst Case**
  - Thereac-25 radiation monitoring devices could not handle high beam current and gave indication of low dosage
- **Failure to Eliminate Root Causes**
  - Focusing on particular software design errors is not a way to make a system safe
  - Tendency to believe that the cause of an accident had been determined without adequate evidence
    - "Patch" one causal factor and assume future will be eliminated
  - Protection against software errors should be built into both the system and software

24 November 2009 13

Lessons Learned

- **Unrealistic Risk Assessment**
  - Assuming that all software errors were equally likely
  - After first incident, no investigation
- **Inadequate Investigation or Follow-up on Accident Reports**
- **Inadequate Software Engineering Practices**
  1. Software specifications and documentation should not be an afterthought
  2. Rigorous software quality assurance should be established
  3. Designs should be kept simple
     - Complex design may be untestable

24 November 2009 14

Lessons Learned

- **Inadequate Software Engineering Practices**
  1. Dangerous coding practices should be avoided
  2. Error detection features should be designed in from the beginning
  3. Extensive testing and formal analysis
  4. Regression testing after each software change
  5. Operator display and user manual need to be carefully designed
  6. Software Reuse
     - Naïve assumption that reusing software will increase safety
  7. Safe operation versus Friendly User Interface
     - Assuming that operators would "double check"
     - Making the machine as easy as possible to use may conflict with safety goals

24 November 2009 15