

# CS 4001: Computing, Society & Professionalism



Munmun De Choudhury | Assistant Professor | School of Interactive Computing

Class 1: Case Study: Therac-25  
January 12, 2017

# The Context

- Radiation therapy
  - Many people with cancer were diagnosed and treated, but were also exposed more radiation than they needed

# Example Case 1


- Therac-20 had optional PDP-11 control, plus built-in hardware checks
- The new technology, Therac-25 relied on software only
- 11 installed machines; 6 major accidents; 3 deaths
  - Improper scanning of the spread of the radiology beam, causing radiation burn and secondary cancer
  - Software errors showing dose was not delivered, technician failed to verify
- Denial – manufacturer and operation refused to believe that the system could make a mistake


# Example Case 2

- The radiation software required that three essential programming instructions be saved in sequence: first, the quantity or dose of radiation in the beam; then a digital image of the treatment area; and finally, instructions that guide the multileaf collimator.
- When the computer kept crashing, the medical physicist, did not realize that her instructions had not been saved.
- It was customary — though not mandatory — that the physicist would run a test before the first treatment to make sure that the computer had been programmed correctly. But the hospital had a staffing shortage.

# Example Case 3

- One therapist mistakenly programmed the computer for “wedge out” rather than “wedge in,” as the plan required.
- Another therapist failed to catch the error.
- And the physics staff repeatedly failed to notice it during their weekly checks of treatment records.

- 
- A Philadelphia hospital gave the wrong radiation dose to more than 90 patients with prostate cancer — and then kept quiet about it.
  - A Florida hospital disclosed that 77 brain cancer patients had received 50 percent more radiation than prescribed because one of the most powerful — and supposedly precise — linear accelerators had been programmed incorrectly for nearly a year.



Dr. Howard I. Amols, chief of clinical physics at Memorial Sloan-Kettering Cancer Center in New York: “Linear accelerators and treatment planning are enormously more complex than 20 years ago. But hospitals are often too trusting of the new computer systems and software, relying on them as if they had been tested over time, when in fact they have not.”

# Why Detection is Difficult

- Identifying radiation injuries can be difficult.
- Organ damage and radiation-induced cancer might not surface for years or decades, while underdosing is difficult to detect because there is no injury.
- For these reasons, radiation mishaps seldom result in lawsuits, a barometer of potential problems within an industry.



# Computerization of Radiation Technology

- Computerization reduced human time needed to calibrate machines and perform safety checks
- But human intervention was still needed to check whether the technology's software came up with a good treatment solution for a patient

# People involved in the tragedies

- Company who made the softwares for the accelerometers
- Programmers and testers behind the softwares
- Doctors who prescribed medication
- Staff and technicians who managed the accelerometers

# Stakeholders: Class Activity I

- Split into groups, have each come up with:
  - what each stakeholder did
  - what they didn't do
  - what they could have done differently
  
- Defense design

# Causal Factors: Class Activity II

- What kind of regulations and check may be put in place to minimize any of the errors that were reported to occur? What should have happened?
- Main causes:
  - software flaws, faulty programming, safe versus friendly interfaces
  - failure to follow a good quality assurance plan, poor safety procedures
  - inadequate staffing and training
  - excessive trust in the software, user and government oversight and standards

# Automation: Classroom Activity III

- Most of you wouldn't work with technology with life-critical implications. But automation is pervasive.
- When is automation good?
- When is it not good?
- What checks should be in place to ensure automation is safe and reliable?

# Software Reuse: Classroom Activity IV

- Most of you wouldn't work with technology with life-critical implications. But software reuse is pervasive in many applications.
- When is reuse good?
- When is it not good?
- What checks should be in place to ensure reuse is safe and reliable?