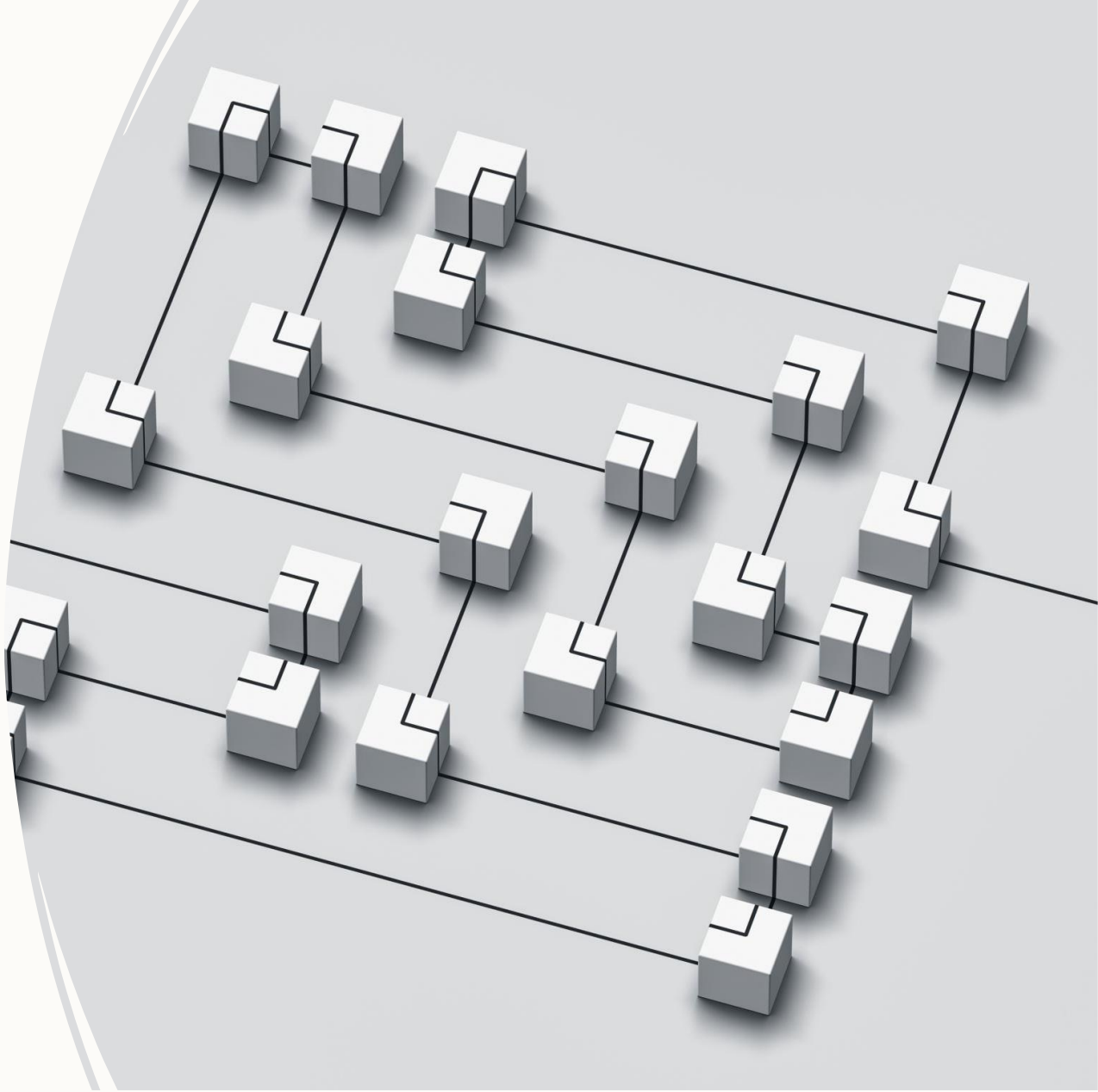
COMP 3056 Professional Ethics in Computing

Week 9 Trust, Safety and Reliability

Learning outcomes

- Causes of Computer Failure
- Bugs and Public Safety
  - When computer failure can lead to harm or death
  - Accountability and Moral Responsibility
- Malware



University of Nottingham
UK | CHINA | MALAYSIA

Trusting computer systems

Trust: Example One (Addition)

Scenario: You want the sum of 1,000 numbers. You get **three different answers** from **three different sources**:

a) One person using **pencil and paper**

b) One person using a **calculator**

c) One person using dedicated **spreadsheet software**

Qu: Which would you be most likely to trust?

Trust: Example Two (Translation)

Scenario: You need a Russian poem translated. You get **three different answers** from **three different sources**:

a) You ask a friend, who is a **Russian major**, for a translation

b) You use dedicated **translation software** to do it yourself

c) You search the poem's title and **find a translation on the Internet**

Qu: Which would you be most likely to trust?

Trust

In some cases we are **right to trust** computers, in other cases we are **wrong**…

Today, we will:

- Explore ways in which people **trust** computers
- Examine how and why computers sometimes **fail** to justify our trust
- Consider the **consequence** of such failure
- Discuss ways to **minimize failure** likelihood
- Discuss **moral Responsibility** / **accountability** for failures

University of Nottingham
UK | CHINA | MALAYSIA

Computer Failure: Causes

Causes of Computer Failure

- Hardware errors
  - Hardware malfunctions causing erroneous results
- Software errors
  - Software "bugs" in code or software design
- Computer solves the wrong problem
  - Programmers incorrectly understand requirements of clients. Program does as intended (no bugs) but does not do what client wants **and expects**
- Misuse
  - Using system for the wrong purpose, or using system incorrectly for the right purpose. E.g., "Garbage in, garbage out"
- Communication failure (in human interaction)
  - User prompted for action misunderstands (e.g., airplane refuel, user assumes amount in gallons rather than litres – plane runs out of fuel!)
- Malice
  - Malicious software (virus, trojan, hacking …) for profit, terrorism, warfare, "fun"

Case: Software Error

- ESA unmanned space flight: Rockets explode ($3.5mn loss)
- Preventable: developers gamble error won't occur
  - Reuse of Ariane 4 code. Floating point (FP) numbers (used for large numbers) converted to integers
  - Ariane 5 accelerated much faster than Ariane 4, so FP values grow much quicker
  - Large 64-bit FP value converted to 16-bit integer. Value couldn't fit, causing arithmetic overflow error
  - Of 7 variables, only 4 protected by error-handling code. Other variables on the "to-do" list!

Case: Hardware Error

- Prof Nicely notified Intel about flaw in P5 chip
  - Occasional flaw in division of one floating point (FP) by another
  - Rare (1 : 9,000,000,000) and unlikely to be noticed by most users
  - But, number of calculations on all PCs in the world is huge!
- Intel already aware of bug, but argued: not important / affects few users
  - Bug was reported by CNN and became “news”
  - Intel offered to replace chips if users demonstrate a need for high precision
  - Public outrage: Intel had obligation to deliver working product; users should not have to predict future usage; users should be able to trust computers deliver precision defined by standards
- Result: Intel agreed to replace all chips
  - Cost: $500mn. Loss of public’s trust and reputational damage

University of Nottingham
UK | CHINA | MALAYSIA

Bugs and Public Safety

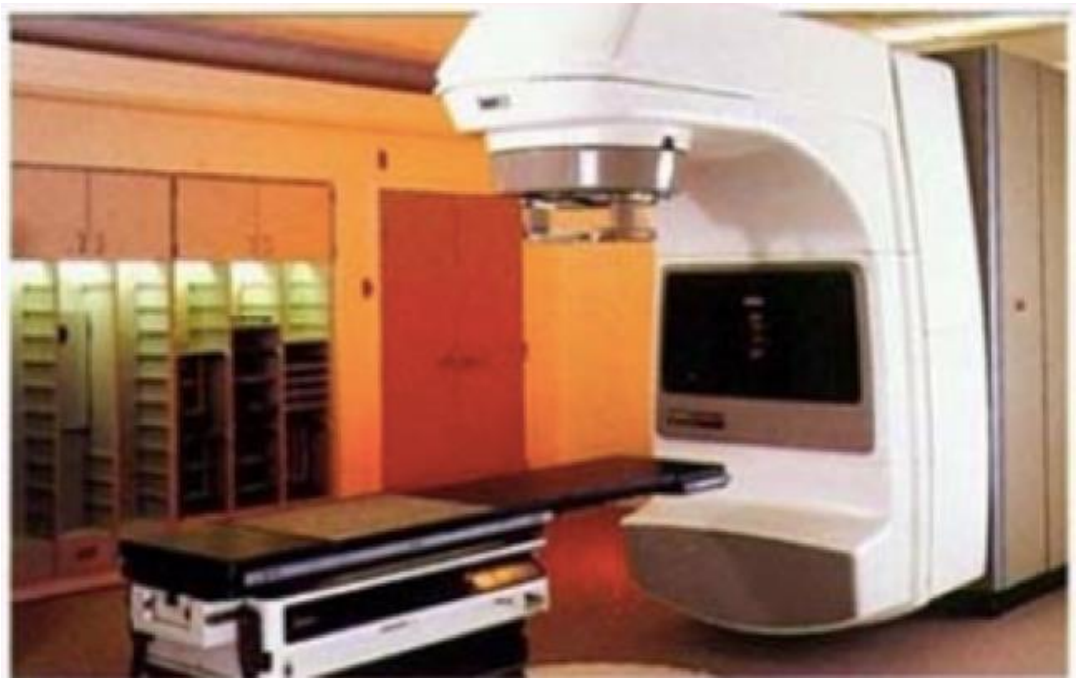
- When computer failure can lead to harm or death
- Accountability and Moral Responsibility

Safety-Critical Software (SCS)

- Software that affect somebody's safety if fails to work properly
  - Air traffic control, nuclear safety system, medical, etc.
- Standards of creating, testing, and maintaining SCS are much stricter than other software
- Exhaustive testing is the obvious answer, but large number of decision-points in software makes this impossible
- Software also control machinery and equipment, therefore
  - Real-time software systems (e.g., avoid object before crash)
  - Software executes in multi-process environments (e.g., accelerate, brake, avoid)
  - These greatly complicate testing

Towards Better Safety-critical Software

- Some improvements have been achieved
  - Better education and training of engineers, programmers, analysts
  - More sophisticated software testing procedures
    - May include **formal verification** (like a mathematical proof)
  - Better maintenance
  - Refocus on prevention rather than detection/correction
  - Simulation can be beneficial (e.g., air-disaster)
  - Emphasis on ethics and professional education



CASE: THERAC-25 (1985-87)

- Machine designed for cancer treatments, known as a medical linear accelerator
- “25” refers to power, much higher than previous machines in series
- Heavy reliance on software for control (and safety)
- Six serious accidents, resulting in 3 deaths (>1,000x required dose!)
- Nancy Leveson, (1995) *Safeware: System Safety and Computers*, Addison-Wesley

Summary of Accidents

(1) 1985: Woman undergoing radiation therapy for breast cancer. Suffered severe burn to shoulder

- Patient reported severe pain
- Physicians didn't "believe" machine burned her
- One recommended applying topical cream
- Radiation burns take days/weeks to present
- Patient contacted lawyers some time later
- Before lawsuit finished, patient died in car accident

USA: Food and Drugs Administration (FDA) www.fda.gov/

Summary of Accidents

(2) Two months later. Woman treated for carcinoma of the cervix. Suffered severe radiation burn to hip

- Patient died shortly after
- Autopsy showed cause of death cancer, not radiation
- Had she lived, hip would need replacing due to burns

(3) December: Radiation overdose causes severe burn

- Patient still living 10 years later

(4,5,6) Three fatal overdoses

Accountability & Moral Responsibility

Douglas Birsch (ethicist and author) analysed the case

- The following people bear moral responsibility
  - The programmer who designed software
  - The company's quality assurance officer
- **Not** legal responsibility
  - That is a matter for the courts to decide

Birsch's Moral responsibility

3 criteria **must be met** for person to bear **moral responsibility**

1. "The person's **actions must have caused the harm**, or been a significant causal factor
2. The person must **have intended or willed the harm**, or it must be a result of his/her **negligence, carelessness, or recklessness**
3. The person must be able to have known, or must **know the consequences of the action**, or must have **deliberately remained ignorant** of them"

Qu: Do these criteria make the programmer(s) morally responsible?

Moral responsibility of the programmer

- Birsch's argument:
  1. The programmer made, or at least allowed, the software errors that directly caused radiation overdose
  2. As software expert, he ought to know that complex software could have bugs unlikely to be detected by testing. He should have notified management of this fact
  3. Programmer knew machine potentially lethal and understood possible consequences of software failure

Therac-25: Lessons learned

- AECL did not have the **software code independently reviewed**
- AECL did not consider the design of the software during its assessment of how the machine might produce the desired results and what failure modes existed. These form parts of the general techniques known as **reliability modeling and risk management**
- The system noticed that something was wrong and halted the X-ray beam, but merely displayed the word "MALFUNCTION" followed by a number from 1 to 64. **The user manual did not explain or even address the error codes,** so the operator pressed the P key to override the warning and proceed anyway.
- AECL personnel, as well as machine operators, initially did not believe complaints. This was likely due to **overconfidence**.
- AECL had **never tested** the Therac-25 with the combination of software and hardware until it was assembled at the hospital.

Now software engineering methods have matured to hopefully stop many of these problems

https://en.wikipedia.org/wiki/Therac-25

University of Nottingham
UK | CHINA | MALAYSIA

Malware

Malware

- **Worm**—makes copies of itself and propagates through network to infect other computers
- **Virus**—similar to worm, but resides in another program. Program must execute for virus to propagate
- **Spyware**—program secretly installed for purpose of collecting information about computer's user(s)
- **Trojan Horse**—software masquerades as innocent and perhaps useful, but actual malicious
- **Rootkit**—program that embeds into OS and acquires special privileges. Potential to override authority of OS, gaining access to passwords and all files

“Ethical” Hacking

- “Hackers”: people who write and deploy malware
- “Malware” indicates malice, so directly immoral
  - To vandalise other’s property, steal from their credit card, or spy on a person, etc. is clearly immoral
- Some “white hat” hackers have legitimate reasons
  - To seek vulnerabilities in system before malicious attacker finds them
  - Risk: how do we know they are not doing illegitimate things? Or, what happens if person was originally convicted of illegal hacking?
  - Sometimes ethical hacking violates the law – you need legal authority
  - Education & training of ethical hackers is controversial
- Zero-day attack
  - Attack that exploits previously unknown vulnerability
- Humans often a weak link in system security
  - Phishing, tricking users, listening devices & cameras

University of
Nottingham
UK | CHINA | MALAYSIA

Summary

Summary

- We all trust computing systems in our daily lives
  - With our health, safety, banking, security, …
- Failure can affect us all and cause pain / death
  - Safety-critical systems
- Failure can be minimized
  - Good software engineering / testing
  - Professional and ethical training
- As professionals, we may bear moral responsibility

Additional Reading

- Chapter 5 of the book *Ethics in a Computing Culture* (Brinkman and Sanders, 2013)
- Chapter 8 of the book *A Gift of Fire* (Baase, 2013)
- Chapter 8 of the book *Ethics for the Information Age* (Quinn, 2013)
- Chapter 6 of the book *Ethics and Computing* (Bowyer, 2001)
- An Investigation of the Terac-25 Accidents. N.G. Leveson, G.S. Turner. *IEEE Computer*, Vol. 26, No. 7, pp. 18-41, 1993.
- *Exploratory Engineering in AI*. Luke Muehlhauser and Bill Hibbard. Communications of the ACM, Vol. 57, No. 9, pp. 32-34, September 2014.

Workshop

- Case study.
- Investigate causes of software failure and responsibility.