CSE312/ISE312

Read Chapter 8

Slides prepared by Cyndi Chie, Sarah Frye, and Sharon Gray. Revisions by R. Kelly

What We Will Cover

- Failures and Errors in Computer Systems
- Case Study: The Therac-25
- Increasing Reliability and Safety
- Dependence, Risk, and Progress
Errors in Computer Systems

- Most computer applications are so complex it is virtually impossible to produce programs with no errors
- The cause of failure is often more than one factor
- Computer professionals must study failures to learn how to avoid them
- Computer professionals must study failures to understand the impacts of poor work

Thematic Issues

- Software engineering licensing
- Does computer science education focus enough on system reliability?
- Inattention to software engineering
  - Documentation
  - Design
  - Testing
- Importance of “first to market”
Problems for Individuals

- Billing errors
- Inaccurate and misinterpreted data in databases
  - Large population where people may share names
  - Automated processing may not be able to recognize special cases
  - Overconfidence in the accuracy of data
  - Errors in data entry
  - Lack of accountability for errors

System Failures

- Many examples cited in text, including
  - Amtrak
  - Voting systems (2000 and 2016 elections)
  - Denver Airport
    - Baggage system failed
    - Main causes:
      - Time allowed for development was insufficient
      - Changes in specifications after the project began
  - Airports in Hong Kong and Kuala Lumpur
Reasons for Systems Failures

- Lack of clear, well-thought-out goals and specifications
- Poor management and poor communication among customers, designers, programmers, etc.
- Institutional and political pressures that encourage unrealistically low bids, low budget requests, and underestimates of time requirements
- Use of very new technology, with unknown reliability and problems
- Refusal to recognize or admit a project is in trouble
- Poor work quality
- Little thought to the lifetime of a system

How much of your CSE education has dealt with long-term implication of software and systems you build?

Design & Development Problems

- Inadequate attention to potential safety risks
- Interaction with physical devices that do not work as expected
- Incompatibility of software and hardware, or of application software and the operating system
- Not planning and designing for unexpected inputs or circumstances
- Confusing user interfaces
- Insufficient testing
- Reuse of software from another system without adequate checking
- Overconfidence in software
- Carelessness
Examples

- Reuse of software: the Ariane 5 rocket and “No Fly” lists
  - It is essential to reexamine the specifications and design of the software, consider implications and risks for the new environment, and retest the software for the new use.
- Space shuttle
  - Schedule pressure
  - Limited testing in fringes of launch parameters

Case Study: The Therac-25

Therac-25 Radiation Overdoses

- Massive overdoses of radiation were given; the machine said no dose had been administered at all
- Caused severe and painful injuries and the death of three patients
- Important to study to avoid repeating errors
- Manufacturer, computer programmer, and hospitals/clinics all have some responsibility
Case Study: The Therac-25

Software and Design problems
- Re-used software from older systems, unaware of bugs in previous software
- Weaknesses in design of operator interface
- Inadequate test plan
- Bugs in software
  - Allowed beam to deploy when table not in proper position
  - Ignored changes and corrections operators made at console

Why So Many Incidents?
- Hospitals had never seen such massive overdoses before, were unsure of the cause
- Manufacturer said the machine could not have caused the overdoses and no other incidents had been reported (which was untrue)
- The manufacturer made changes to the turntable and claimed they had improved safety after the second accident. The changes did not correct any of the causes identified later.
Case Study: The Therac-25

Why So Many Incidents? (cont.)

- Recommendations were made for further changes to enhance safety; the manufacturer did not implement them.
- The FDA declared the machine defective after the fifth accident.
- The sixth accident occurred while the FDA was negotiating with the manufacturer on what changes were needed.

Observations and Perspective

- Minor design and implementation errors usually occur in complex systems; they are to be expected
- The problems in the Therac-25 case were not minor and suggest irresponsibility
- Accidents occurred on other radiation treatment equipment without computer controls when the technicians:
  - Left a patient after treatment started to attend a party
  - Did not properly measure the radioactive drugs
  - Confused micro-curies and milli-curies
Professional Techniques

- Importance of good software engineering and professional responsibility
- User interfaces and human factors
- Redundancy and self-checking
- Testing
  - Include real world testing with real users
- Management and communication
- High reliability organization principles
  - preoccupation with failure
  - loose structure

What would your approach be towards testing a new cloud computing service?

Specifications

- Learn the needs of the client
- Understand how the client will use the system
UIs and Human Factors

- User interfaces should:
  - provide clear instructions and error messages
  - be consistent
  - include appropriate checking of input to reduce major system failures caused by typos or other errors a person will likely make
- The user needs feedback to understand what the system is doing at any time.
- The system should behave as an experienced user expects

How do you define a range of user errors in the testing phase?

Corresponding page number: 385

Redundancy and Self-Checking

- Multiple computers capable of same task; if one fails, another can do the job.
- Voting redundancy

Is it usually possible to provide for human/manual system backup?

Corresponding page number: 386-387
Testing

- Even small changes need thorough testing
- Independent verification and validation (IV&V)
- Beta testing

Is robust testing a typical part of your SBU projects?

---

Law, Regulation, and Markets

- Criminal and civil penalties
  - Provide incentives to produce good systems, but shouldn't inhibit innovation
- Regulation for safety-critical applications
- Professional licensing
  - Arguments for and against
- Taking responsibility

Is the threat of litigation an effective incentive for software developers to build safer and more reliable systems?

---