CS305 Topic – Reliability

- Errors in Computer Systems
- Impacts of Computer Errors
- Lessons Learned
- How to Improve?

Sources: Baase: A Gift of Fire *and* Quinn: Ethics for the Information Age

Errors in Computer Systems

- Data-related errors
  - Erroneous information in databases
  - Misinterpretation of database information
- Software errors
- System failures

Effects of Computer Errors:

- Inconvenience
- Financial loses
- Fatalities

Data Error Example

November 2000 general election, Florida disqualified thousands of voters.

- *Reason:* People identified as felons
- *Cause:* Incorrect records in voter database
- *Consequence:* May have affected election's outcome

False Arrests

Due to Inaccuracy in NCIC Records:

- Sheila Jackson Stossier mistaken for Shirley Jackson
  - Arrested and spent five days in detention
- Roberto Hernandez mistaken for another Roberto Hernandez
  - Arrested twice and spent 12 days in jail
- Terry Dean Rogan arrested after someone stole his identity
  - Arrested five times, three times at gun point

Who Should Be Responsible?

Privacy Act of 1974:

"Each agency … shall … maintain all records … with such accuracy, relevance, timeliness, and completeness as is reasonably necessary to assure fairness to the individual in the determination"

Privacy Advocates:

- Number of NCIC records is increasing (> 40 mil)
- Accuracy of records is more important than ever
- Government must fulfill its responsibility

Justice Dept's Position

Impractical for FBI to be responsible for data's accuracy:

- Much information provided by other law enforcement and intelligence agencies; hard to verify
- If full accuracy is required, much less information would be in NCIC, making it less useful

March 2003:

Justice Dept. announces FBI not responsible for accuracy of NCIC records; Exempts NCIC from some provisions of the Privacy Act of 1974.

Software Error Examples

- U.S. Postal Service returns 50,000 mail addressed to Patent and Trademark Office (1996)
- Qwest sends incorrect bills to 14,000 cell phone customers (2001)
- Amazon.com in Britain offered iPaq for £7 instead of £275; Amazon.com shut down site, refused to deliver unless customers paid true price (2003)

Question:

Was Amazon wrong to refuse to fill the orders?

Hospital Lab Computer System

- A Medical Center in LA
- Computer Failure

"It's almost like practicing Third World medicine. We rely so much on our computers and our first-world technology that we were almost blinded."

--- A ER Doctor

Amazon Case Analysis

Utilitarian Analysis:

- *Proposed Rule:*

  A company must always honor the advertised price.

- *Consequences:*
  - Companies would spend more time proofreading ads, and may take out insurance policies
  - Higher costs → higher prices for all consumers
  - Only few customers would benefit from errors
- *Conclusion:*
  - Rule has more harms than benefits
  - Amazon.com did the right thing

Amazon Case Analysis (cont.)

Kantian Analysis:

- Buyers knew 97.5% markdown was an error
- They attempted to take advantage of Amazon.com's stockholders
- They were not acting in "good faith"
- Buyers did something wrong

Notable System Failure Cases

- Therac-25 (1985-86)
- Airbus A320 (1988-92)
- AT&T long-distance network (1990)
- Patriot missile (1991)
- Denver international airport (1993)
- Ariane 5 (1996)
- Robot missions to Mars (1999)

Several of these failures caused fatalities.

Therac-25

- Linear electron-beam/x-ray accelerator for medical use (built by AECL)
  - First model with an integrated computer (PDP-11)
  - Hardware safety features replaced with software
  - Reused code from Therac-6 and Therac-20
  - First Therac-25 shipped in 1983
- Between 1985-1987, six patients were given massive overdoses of radiation (100x); three died

Therac-25: Chronology

- *June 1985* – Case #1 (Marietta, Georgia)
- *July 1985* – Case #2 (Hamilton, Ontario)
- *July-Sept. 1985* – First AECL investigation
  - "Can't reproduce the overdose"
- *Dec. 1985* – Case #3 (Yakima, Washington)
- *Mar. 1996* – Case #4 (Tyler, Texas)
- *Mar. 1996* – Second AECL investigation
  - Still can't reproduce the overdose
- *Apr. 1986* – Case #5 (Tyler, Texas)
- *Jan. 1987* – Case #6 (Yakima, Washington)
- *Feb. 1987* – FDA declares Therac-25 defective

Therac-25 Design Flaws

- Re-used software from older systems, unaware of bugs in previous software
- The software was not independently reviewed or tested – in fact, the software was mostly developed and tested by one single person
- System not designed to be fail-safe
  - No devices to report overdoses
  - No way for patient to communicate with operator during procedure
- Weaknesses in design of operator interface

Therac-25 Software Bugs

- Allowed beam to deploy when table not in proper position
- Ignored changes and corrections operators made at console
- Race conditions

Airbus A320

- A320 are called "fly-by-the-wire" airplanes – many systems are controlled by computers; not directly by the pilots
- Between 1988-1992 four planes crashed

Causes:

- Conflicts between pilots and computers
  - The airplane has “a mind of its own”
- Computer errors
  - Failed to detect landing

AT&T Long-Distance Network

- About half of routing switches crashed
- 70 million calls not put through
- 60,000 people lost all service
- AT&T lost revenue and credibility

Cause:

- Single-line error in error-recovery procedure
- Most switches running same software
- Crashes propagated through the network

Patriot Missile

- Used in Gulf War to intercept Scud missiles
- One battery failed to shoot down a Scud that killed 28 soldiers

Cause:

- Designed to operate only a few hours at a time
- Kept in operation > 100 hours
- Tiny truncation errors added up
- Clock error of 0.3433 seconds → tracking error of 687 meters

Scud Missile
(wikipedia photo)

Patriot Antimissile Defense System
(wikipedia photo)

Denver International Airport

- The one-of-a-kind, state-of-the-art automated baggage handling system failed to work
  - 16-month delay in opening the airport
  - Cost Denver $1 million a day

Problems:

- Airport designed before automated system chosen
- System complexity exceeded developer's ability
- Timeline too short

Fix: Added conventional baggage system

Ariane 5

- Satellite launch vehicle
- 40 seconds into maiden flight, rocket self-destructed, $500 million of satellites lost

Cause:

- Statement assigning floating-point value to integer raised exception
- Exception not caught and computer crashed
- Code reused from Ariane 4
  - Slower rocket, smaller values being manipulated

Robot Missions to Mars

- Climate Orbiter disintegrated in Martian atmosphere

  Cause:

  - Lockheed Martin design used English units
  - Jet Propulsion Lab design used metric units
- Polar Lander crashed into Martian surface

  Cause:

  - False signal from landing gear, causing engines shut off too soon

Summary of Failure Causes

Patriot

Equipment was not operated to its exact specifications.

Operator Error or *Design Error*???

Ariane 5

Software designed for one application is moved to another application for which the parameters were slightly different. New team fails to appreciate this.

ATT Long Distance

Unforeseen Emergent Behavior of a complex system.

Summary of Failure Causes

Mars Climate Orbiter

Multiple teams failed to communicate clearly.

Mars Polar Lander

False sensor signal (?)

Denver Airport Baggage System

Design timeline was too short.

System was too complex.

Summary of Failure Causes

Technical:

- Use of very new technology, with unknown reliability and problems
- Reuse of software, without adapting to new conditions
- Lack of clear, well thought out goals and specifications
- Lack of thorough testing

Managirial:

- Poor management and poor communication among customers, designers, programmers
- Pressures that encourage unrealistically low bids and underestimates of time requirements
- Refusal to recognize or admit project problems

Who Is Responsible?

- Software developers?
- Software vendors?
- System administrators?

Question:

- If you were a judge who had to assign responsibility in Therac-25 case, how much responsibility would you assign to the programmer, the manufacturer, and the hospital or clinic using the machine?

How to Improve Reliability?

Difficulties:

- Software complexity
- Software is only part of a system
- Formal verification tools still immature

Directions:

- Solid software engineering practice
- Regulation on safety-critical applications
- Professional licensing(?)

Software Complexity

Examples:

- Linux kernel 2.6.0 – 6 million lines of code
- Redhat Linux 7.1 – 30 million lines of code
- Windows XP – 40 million lines of code

In comparison:

- Boeing 747 – 3.5 million parts
- Space shuttle – 10 million parts

The point:

Formal methodology is needed for software development

Software is Only a Part

Computer simulations replace physical experiments in many fields:

- Experiment too expensive/time-consuming
- Experiment unethical
- Experiment impossible
- Can model past, current, and predict the future

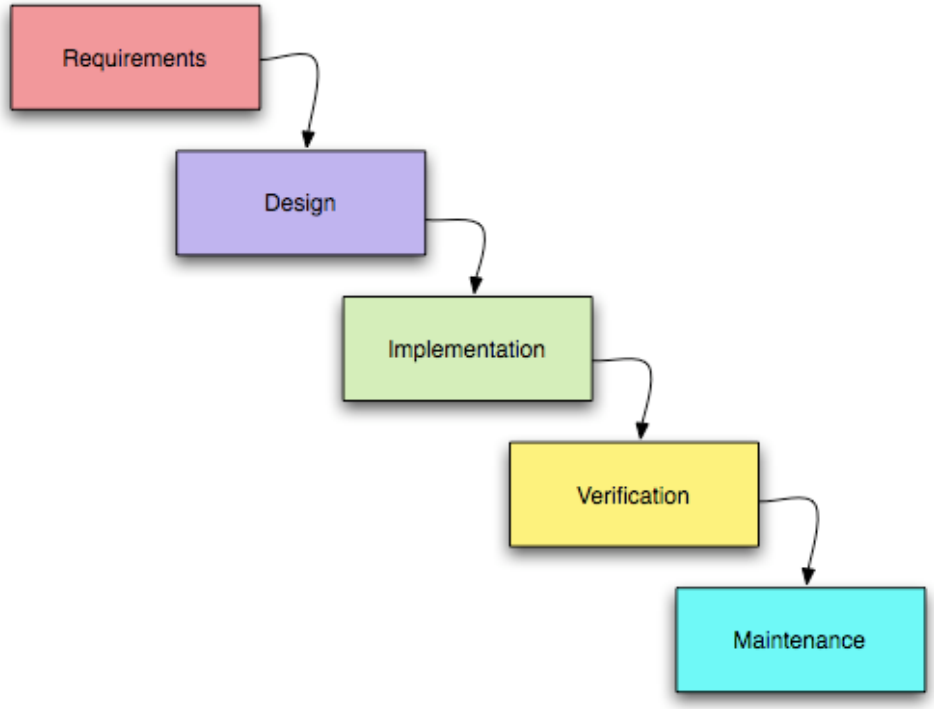
However, accuracy and reliability is only as good as the weakest link:

- Does the model accurately represent the real system?
- Does program correctly implement the model?

Software Engineering Principles

The Waterfall Model:

1. *Requirements*
2. *Design*
3. *Implementation*
4. *Verification*
5. *Maintenance*



Key Idea:

Have a fixed comprehensive requirement specification and stick with it to the end.

Software Quality Is Improving

Standish Group tracks IT projects:

- Situation in 1994
  - 1/3 projects cancelled before completion
  - 1/2 projects had time and/or cost overruns
  - 1/6 projects completed on time on budget
- Situation in 2002
  - 1/6 projects cancelled
  - 1/2 projects had time and/or cost overruns
  - 1/3 projects completed on time on budget

Software Warranties

- Many are “shrink-wrapped” – can’t read before purchasing.

  Question: Are “shrink-wrapped” agreements legally enforceable?

- Most say you accept software “as is”. *None* accept liability for harm caused by use of software.

  Question: Can software manufacturers choose any warranty terms they want on their products?

Notable Court Cases

- ProCD, Inc. *v.* Zeidenberg
  - Zeidenberg purchased a ProCD software CD; made copies and resold the copies, claiming he did not see the license.

  Court Rule: Shrink-wrap licenses are enforceable
- Mortensen *v.* Timberline Software
  - Mortensen purchased buggy Timberline software that caused financial loss.

  Court Rule: Liability limits in warranties are enforceable

Scenario

A software vendor sell some buggy software.

Later the bugs are fixed…

…and the fixes are incorporated into next version.

The new version is available, but to get it, the users much purchase it.

Is it ethical for the company to force users to purchase a new version in order to get their bugs fixed?

Consumer Protection Laws

- Magnuson-Moss Warranty Act (1975):
- Requires manufacturers/sellers to provide consumers with detailed information about warranty coverage.
- It defines the rights of consumers and the obligations of warrantors under written warranties.

Problem:

Only applicable to *full* warranties. Yet, no requirement on what type of warranty manufacturers use.

- Article 2 ("Sales") of Uniform Commercial Code (enacted in 1960s, updated in 2003)

UCITA

A proposed law to create a uniform set of rules to govern transactions in computer information (e.g. software licensing and online access).

Under UCITA, software manufacturers can

- License software
- Prevent software transfer
- Disclaim liability
- Remote disable licensed software
- Collect information about how software is used

UCITA (cont.)

Arguments in Favor:

- Article 2 of the UCC not appropriate for software
- Recognizes there is no such thing as perfect software

Arguments Against:

- Customers should be allowed to *purchase* software
- Weakens consumer protections
- Codifies practice of hiding warranty

UCITA is unlikely to pass without amendments.

Discussion Questions

- Have you been the victim of a software error? Whom did you blame? Now that you know more about the reliability of computer systems, do you still feel the same way?

- You are in charge of developing a software system that controls the traffic lights. Its main purpose is to adjust the timing of the lights to improve traffic flow at rush hours. List some technical requirements that you would put in the design for safety.

Discussion Questions

- Should software manufacturers be responsible for harmful consequences of defects of their products?

- Software companies sometimes release bug-fixes on their product to their customers free of charge. However, they often stop doing it when a new version of the product is released. Do you think this practice is fair? Or should companies keep fixing bugs in older versions?