Social Implications of a Computerized Society

Computer Errors

Instructor: Oliver Schulte

Simon Fraser University

Failures and 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

Discussion Question

- Have you personally suffered from computer system errors? What should change to prevent these errors?

Errors and consequences

Error Types

- Data Entry Errors
  - Along with privacy, another concern about large databases.
  - Recommended principle: give people option to view and correct information.
- **Crashes and Safety-Critical Applications**
- Performance Failures: Estimated 5-15% of IT projects are abandoned soon before or after delivery as “hopelessly adequate”

Data Errors

- Inaccurate and misinterpreted data in databases
- Billing errors
- no-fly lists: Joseph Adams on list ➔ J(ane) Adams could not board
- Causes
  - Large population where people may share names
  - Reuse of previous name match software
  - 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

- AT&T, Amtrak, Skype
- Businesses have gone bankrupt after spending huge amounts on computer systems that failed
- Ariana 5 Rocket: went off-course and was destroyed after 40 seconds.
  - float overflow because faster Ariana 5 faster than Ariana 4

Denver Airport:

- Baggage system failed due to real world problems, problems in other systems and software errors
- Delay cost $30 Million/month.

Causes

- Main causes:
  - Time allowed for development was insufficient
  - Denver made significant changes in specifications after the project began
- Great Reference: The Mythical Man-Month
  - Available for free to SFU students on-line

safety-critical applications

Safety-Critical Applications

- Air traffic control is extremely complex, tracking airplanes that move very fast
- Trend towards autonomous planes (TCAS) computers avoid collisions, not pilots
- In spite of problems, computers and other technologies have made air travel safer
- ACM maintains a RISK update

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

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

Lack of Reactions

- 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
- Similarities to Boeing 737 Max story

Discussion Questions

- Have we become too dependent on computers? Should we use them less?

software complexity and specifications

Sources of Complexity

- Computer is doing a difficult job.
- “Non-linearity” (discontinuity):
  - In physical device, small error usually makes small difference in performance.
  - In computer program, small typo can make big difference.

Increasing Reliability and Safety

Professional techniques:

- Importance of good software engineering and professional responsibility
- unit tests! Break systems into reliable pieces
- User interfaces and human factors
  - Feedback
  - Should behave as an experienced user expects
  - Workload that is too low can lead to mistakes
- Redundancy and self-checking
- Testing
  - Include real world testing with real users

Understand the specs

- Specs = specifications, desired performance.
- What is the problem the user wants to solve? What are the circumstances under which the system will be deployed?

Example for Specification Issues

- Automated airport luggage system failed in several airports (Kuala Lumpur, Detroit).
  - One problem was designers did not allow for data entry errors.
  - Also hardware failures (scanners).
- MS handwriting system did not work for Bill Gates – he is left-handed!
- Newborn baby weighing system rounded off ounces---important for premature babies!

Ways to get the specs right

- Testing, including beta testing by community.
- Consult with users.
- Write down your assumptions. (Ideally in the code.) Can you prove correctness?
- Formal Verification. (Turing Award 2008!)

Practical Problems: Specs

- Too little training in vague specs - assignments have strictly defined instructions.
- Problems with Testing.
  - Confirmation bias: People tend to look for ways to confirm their system works, not for ways to make it crash
    - 3rd-party testing
  - Expensive in money and time.
- Problems with User Consultation.
  - Access to users can be limited (managers, experts).
  - Single demo for users different from real-life, continuous use.
  - Users resist new technology.

What Can Go Wrong?

- Consider issues of safety, risk, privacy.
- Do not rely on customization---users may not bother to change defaults.
- Be open about capabilities and limitations of your system.
- In practice:
  - Pressure to cut corners, and to sell.
  - Not easy to communicate risks to users (cf. medical treatment side effects).

Communicating with Users

- Explain product/design choices, security risks.
- Give clear, interesting, well-prepared presentations.
  - Take responsibility for liveliness, getting attention.
  - If a user does not get the relevant info, it is your responsibility too.

Learn to Communicate with Managers

- True story: Programmer asked for meeting with owner of her company. Explains that beta testing should not be skipped and why. Owner accepts, reschedules release date. Programmer becomes head of quality control.

Problems:

- Group Dynamics, e.g. Middle Managers want to save face.
- Honouring Sunk costs: Refusal to admit a project is in trouble
- Software Market does not encourage quality.

NOT ON EXAM

Economic and legal perspectives

Economic Perspectives on the Software Market: The Buyers/Users.

- The Software Market has a lot of what economists call "information asymmetries", i.e. the sellers know things the buyers do not.
- Primary functionality is relatively easy to assess for user (e.g., play music, word processing).
- Reliability, security, risk very difficult to assess.
- In such markets, users focus on:
  - What they can assess, i.e. what does this thing do?
  - Reputation markers, e.g. reviews, brands (MS, HP, IBM etc.)
  - Use by friends, co-workers etc. $\Rightarrow$ networking effects.

Economic Perspectives on the Software Market: The Designers/Companies.

- Software businesses have cash flow issues.
  - Large investment required before product ready for sale (gets worse as complexity increases) $\Rightarrow$ "barrier to market entry".
  - Solutions:
    - Venture capital.
    - Government subsidies/loans.
    - Open Source Development.

Market Incentives for Software Companies

- Reliability, security often **not big selling points** because hard to assess by end user (see above) $\Rightarrow$ market does not reward investment.
- Unreliable systems can actually make **more profit**: keep charging for updates, support (SAP, SIMS?).
- Cash flow problems create pressure to release fast, to get more sales or at least more investment.

A Simple Model of Market Incentives: calculate the profits

Options for business/ user population	20% of users are sensitive to reliability, safety, privacy	80% of users buy if they want the primary function
High quality product, costs $10[5]	Sell each product for $10	$10
Low quality product	$0	$10

Law and Regulation

- Criminal and civil penalties
  - Provide incentives to produce good systems, but should not inhibit innovation
- Warranties for consumer software
  - Most are sold ‘as-is’
- Regulation for safety-critical applications
- Professional licensing
  - Arguments for and against

Discussion Question

1. Should there be mandatory professional licensing for IT professionals (as with engineers?)
2. Should there be more warranty for software (e.g., can sue manufacturer if software causes damage)? For certain kinds of software?

Theme for the Future?

- Are we reaching limits in computer use/scope?
  - Moore's "law" on processor performance is no longer valid.
  - Complex systems (cell phones have several million lines of code), hard to test, expand, integrate.
  - Unsolved tasks are complex?!