Bugs, (un)reliability, errors, risks

Robert H. Sloan

Portions from Lecture notes from *Gift of Fire Lecture* 2nd and 3rd education and Quinn, *Ethics for the Information Age*

The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair.

—Douglas Adams, *Mostly Harmless*

Even without black hats

- System designers make mistakes
- Data entry errors
- Murphy
- Software-based systems inherently daedal

Eastport Report (1985), page 14

Simply because of its inevitable large size, the software capable of performing the battle management task for strategic defense will contain errors. All systems of useful complexity contain software errors.

From Murphy to Satan

- And avoiding security flaws is a harder problem than avoiding ordinary flaws.

Facts about Computer Errors

- Error-free software is not possible
  - But software can have very different rates of error
  - Errors can be reduced by following good software engineering procedures/practices
- Errors are often caused by more than one factor
- Line between tolerable and/or unavoidable errors versus careless software development blurry but exists

My opinions on software errors

- Discrete versus continuous math leads to fewer 9's of reliability.
- 1975 Bridge over roadway not crashing; POTS (Plain-old-telephone system): easily 0.99999 chance that if you picked up phone, you got a dial tone.
- Not true of Windows XP, Vista, nor Windows 7! Nor Mac OS X.

Database entry and/or retrieval errors

- November 2000 general election Florida disqualified 1000s of voters charged with **misdemeanors**; **maybe (probably) changing outcome of Presidential Election**
- False arrests because of errors among 40 million entries in National Crime Information Center (NCIC)
  - But 10,000s of recovered stolen cars too.

DoJ on NCIC records

- March 2003: Justice Dept. announces FBI not responsible for accuracy of NCIC information
- Exempts NCIC from some provisions of Privacy Act of 1974

Database Error Causes

- Large population ⇒ Name collisions
- Human common sense not part of automated processing
- Errors in data entry
- Overconfidence in accuracy of data from a computer
- Information not updated or corrected
- Lack of accountability for errors

Sample errors: less than total failure

- Qwest sends incorrect bills to cell phone customers
- Faulty USDA beef price reports
- U.S. Postal Service returns mail addressed to Patent and Trademark Office
- Spelling and grammar error checkers increased errors
- BMW on-board computer failure
- 9/2008–5/2009: NYC overcharges public housing rent because of error in program computing bills; ignores renters complaints; files eviction against those paying only proper amount

System Failures: A Sample

- Los Angeles County + USC Medical Center laboratory computer, April 2003
- Japan's air traffic control system, March 2003
- Chicago Board of Trade, Jan. 23 and April 1, 1998
- London International Financial Futures and Options Exchange 1999
- Comair's Christmas Day shutdown, 2004

Software Hall of Shame

YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of $50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after $4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after $40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to $3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause $350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after $25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent.
1994	State of California	DMV system canceled after $44 million is spent.
1994	Chemical Bank	Software error causes a total of $15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after $600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing $130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at $11.25 million**; second attempt abandoned after deployment, costing $15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of $61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after $165 million is spent.

Famous System Failures

A small sample of the infamous

AT&T Long-Distance System, 1990

- AT&T disrupted 9 hours; 50–70 million failed calls
- AT&T lost revenue and credibility
- Cause
  - Single line of code in error-recovery procedure; overall program 4 million lines of code
  - Most switches running same software
  - Crashes propagated through switching network
- AT&T Official Report: "While the software had been rigorously tested in laboratory environments before it was introduced, the unique combination of events that led to this problem couldn't be predicted."

Ariane 5

- Ariane 5 satellite launch vehicle (France) 1996; conversion of 64-bit float to 16-bit signed int oveflowed; exception not handled.
  - Loss: $500 million of satellites (uninsured)
- Code reused as-is from Ariane 4, where value was guaranteed to be small enough

Denver International Airport, 1995

- Denver airport opening delayed 4 times, cost of $30 million/month of delay (bond interest & operating costs). Most of delay computer-controlled baggage system, $200 million of $3.2 billion airport.
  - Spec: outbound checked luggage in < 10 minutes on carts at 19 mph on 22 miles of underground tracks. Laser scanners tracking 4000 carts
  - Nobody expects software/hardware system of this complexity to work right first time

Denver fiasco

- Real-world problems: scanners got dirty, knocked out of alignment
- Problems in other systems: Airport electrical system could not handle power surges associated with baggage system; massive circuits blowing
- Software errors (needed carts routed to waiting pens)
- Overall: *Inadequate development and testing time; significant changes in specs after project start*

DIA final outcome

- In June 2005, United Airlines scrapped the trouble-plauged automated baggage system.
- Returned to manual baggage handling after having spent about $200 million
- Return to manual baggage handline estimated to save $1 million/month in system maintenance costs, and more in cossts of misdirected and damaged bags.

When will they ever learn?

- More ambitious plans for Hong Kong and Kuala Lumpur (Malaysia) airports
  - Ambitious, complex computer systems to manage *everything:* moving 20,000 pieces baggage/hour, coordinating and scheduling crews, gate assignments for flights, etc.
- Both failed spectacularly

Tokyo Stock Exchange

- Dec. 8, 2005, J-Com went public
- Mizuho Securities Employee mishears "Sell 1 share at 610,000 yen" as "Sell 610,000 shares at 1 yen."
- Mizuho repeatedly tries to cancel order over 6 minutes; bug in exchange software causes cancellation to fail.
- J-Com had 14,500 publicly traded shares
- Mizuho lost $225 million; litigation between Mizuho and Tokyo Stock Exchange ongoing.
- Bug had existed 5 years; only occurred when 7 unusual conditions all held simultaneously.

System failure causes

- Bugs are hard to find, especially those that occur only in specific situation ("flakey")
- Difficult to debug concurrent systems
- Insufficient testing and debugging time
- Significant changes in specifications (after project begun)
- Overconfidence in system
- Project mismanagement

Saftey-Critical Applications

- Aircraft; Trains
- Military
- Power plants
- Automated Factories
- Medicine
- Etc.

Causes of safety-critical failures

- Overconfidence
- Lack of override features
- Insufficient testing
- Sheer complexity of the system
- Mismanagement

Therac-25 overview

- Therac-25 was a software controlled radiation therapy machine used to treat people with cancer.
- Resulted in overdoses of radiation
  - Normal dosage 100–200 rads.
  - ≥6 people got estimated 13,000–25,000 rads
  - Three of the six known people died.
- Important to study to avoid repeating the errors
- Manufacturer, programmer(s), hospitals/clinics all share some of the blame

Therac-25: multiple causes

- Poor safety design
- Insufficient testing and debugging
- Software errors
- Lack of safety interlocks
- Overconfidence
- Inadequate reporting and investigation of accidents

Therac-25 Software & 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
  - Sometimes ignored changes and corrections operators made at console.
    - Race condition: ironically meant the better, faster operators were the ones likely to kill somebody.

Therac-25: 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

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

Why do software projects fail? (Charette 2005)

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

My take

- Computer Systems fail because:
  - The job they are doing is inherently difficult
  - The job is done poorly
- We do not know how to build perfect, flawless, 0 bugs software
- We do know how to build very good software with few bugs; we often don't
- Compounding the reliability issue: Insufficient market or legal incentives to do a better job

Professional Techniques

- Follow good software engineering practices
- Take human factors into account, especially construct well-designed user interfaces
- Incorporate self-checking where appropriate, redundancy in saftey/wealth-critical systems
- Follow good testing principals and techniques

A little better all the time?

"… large software project failures: it has been known for years that perhaps 30% of large development projects fail [24], and this figure does not seem to change despite improvements in tools and training: people just built much bigger disasters nowadays than they did in the 1970s."

Anderson, Crypto 07 Keynote; similar in 2009 book

Vs. Standish Group, which tracks 1000s of IT Projects

- 1994: 31% failed (cancelled before completion); 16% on time & in budget
- 2006: 19% cancelled, 35% on time & in budget
- 2009: 24% cancelled, 32% on time & in budget

Increasing Reliability?

- Would forced real warranties help?
  - When I was in law school in the Dark Ages, used to be something called UCC, and Magnusun-Moss.
- Professional Licensing of software engineers a current controversial discussion

For more info

- Peter G. Neumann's "Inside Risks" email/web site has been the source where sad and scary stories have been collected since the 1980s.