



Dependable Software Systems

Software Related Risks

“If anything can go wrong, it will
(and at the worst possible moment).”

- Murphy

Material drawn from [Neumann]

Courtesy Spiros Mancoridis



Sources of Problems

- **Requirements Definition:** Erroneous, incomplete, inconsistent requirements.
- **Design:** Fundamental design flaws in the software.
- **Implementation:** Mistakes in chip fabrication, wiring, programming faults, malicious code.
- **Support Systems:** Poor programming languages, faulty compilers and debuggers, misleading development tools.



Sources of Problems (Cont'd)

- **Inadequate Testing of Software:**
Incomplete testing, poor verification, mistakes in debugging.
- **Evolution:** Sloppy redevelopment or maintenance, introduction of new flaws in attempts to fix old flaws, incremental escalation to inordinate complexity.



Adverse Effects of Faulty Software

- **Communications:** Loss or corruption of communication media, non delivery of data.
- **Space Applications:** Lost lives, launch delays.
- **Defense and Warfare:** Misidentification of friend or foe.



Adverse Effects of Faulty Software (Cont'd)

- **Transportation:** Deaths, delays, sudden acceleration, inability to brake.
- **Safety-critical Applications:** Death, injuries.
- **Electric Power:** Death, injuries, power outages, long-term health hazards (radiation).



Adverse Effects of Faulty Software (Cont'd)

- **Money Management:** Fraud, violation of privacy, shutdown of stock exchanges and banks, negative interest rates.
- **Control of Elections:** Wrong results (intentional or non-intentional).
- **Control of Jails:** Technology-aided escape attempts and successes, accidental release of inmates, failures in software controlled locks.
- **Law Enforcement:** False arrests and imprisonments.



Bug in Space Code

- Project Mercury's FORTRAN code had the following fault:
DO I=1.10 instead of ... **DO I=1,10**
- The fault was discovered in an analysis of why the software did not seem to generate results that were sufficiently accurate.
- The erroneous 1.10 would cause the loop to be executed exactly once!



Military Aviation Problems

- An F-18 crashed because of a missing exception condition:
if ... then ... without the **else** clause that was thought could not possibly arise.
- In simulation, an F-16 program bug caused the virtual plane to flip over whenever it crossed the equator, as a result of a missing minus sign to indicate south latitude.



Year Ambiguities

- In 1992, Mary Bandar received an invitation to attend a kindergarten in Winona, Minnesota, along with others born in '88.
- Mary was 104 years old at the time.



Year Ambiguities (Cont'd)

- Mr. Blodgett's auto insurance rate tripled when he turned 101.
- He was the computer program's first driver over 100, and his age was interpreted as 1.
- This is a double blunder because the program's definition of a teenager is someone under 20!



Dates, Times, and Integers

- The number $32,768 = 2^{15}$ has caused all sorts of grief from the overflowing of 16-bit words.
- A Washington D.C. hospital computer system collapsed on September 19, 1989, 2^{15} days after January 1, 1900, forcing a lengthy period of manual operation.



Dates, Times, and Integers (Cont'd)

- COBOL uses a two-character date field ...
- The Linux **term** program, which allows simultaneous multiple sessions over a single modem dialup connection, died word wide on October 26, 1993.
- The cause was the overflow of an **int** variable that should have been defined as an **unsigned int**.



Shaky Math

- In the US, five nuclear power plants were shut down in 1979 because of a program fault in a simulation program used to design nuclear reactor to withstand earthquakes.
- This program fault was, unfortunately, discovered after the power plants were built!



Shaky Math (Cont'd)

- Apparently, the arithmetic sum of a set of numbers was taken, instead of the sum of the absolute values.
- The five reactors would probably not have survived an earthquake that was as strong as the strongest earthquake ever recorded in the area.



Therac-25 Radiation “Therapy”

- In Texas, 1986, a man received between 16,500-25,000 rads in less than 1 sec, over an area of about 1 cm.
- He lost his left arm, and died of complications 5 months later.
- In Texas, 1986, a man received at least 4,000 rads in the right temporal lobe of his brain.
- The patient eventually died as a result of the overdose.



Therac-25 Radiation “Therapy” (Cont’d)

- In Washington, 1987, a patient received 8,000-10,000 rads instead of the prescribed 86 rads.
- The patient died of complications of the radiation overdose.



AT&T Bug: Hello? ... Hello?

- In mid-December 1989, AT&T installed new software in 114 electronic switching systems.
- On January 15, 1990, 5 million calls were blocked during a 9 hour period nationwide.



AT&T Bug (Cont'd)

- The bug was traced to a C program that contained a **break** statement within an **switch** clause nested within a loop.
- The **switch** clause was part of a loop. Initially, the loop contained only **if** clauses with **break** statements to exit the loop.
- When the control logic became complicated, a **switch** clause was added to improve the readability of the code ...



Bank Generosity

- A Norwegian bank ATM consistently dispersed 10 times the amount required.
- Many people joyously joined the queues as the word spread.



Bank Generosity (Cont'd)

- A software flaw caused a UK bank to duplicate every transfer payment request for half an hour. The bank lost 2 billion British pounds!
- The bank eventually recovered the funds but lost half a million pounds in potential interest.



Making Rupee!

- An Australian man purchased \$104,500 worth of Sri Lankan Rupees.
- The next day he sold the Rupees to another bank for \$440,258.
- The first bank's software had displayed a bogus exchange rate in the Rupee position!
- A judge ruled that the man had acted without intended fraud and could keep the extra \$335,758!

Dependable Software Systems (Risks)



Bug in BoNY Software

- The Bank of New York (BoNY) had a \$32 billion overdraft as the result of a 16-bit integer counter that went unchecked.
- BoNY was unable to process the incoming credits from security transfers, while the NY Federal Reserve automatically debited BoNY's cash account.



Bug in BoNY Software (Cont'd)

- BoNY had to borrow \$24 billion to cover itself for 1 day until the software was fixed.
- The bug cost BoNY \$5 million in interest payments.