CL1 23:
Getting it right and getting it wrnog

Monday 13/11/2006

Introduction

- Bugs
  - – Definition
  - – Examples
- Algorithms
  - – Foundation of computer programs
  - – All applications are programs
- Software design
  - – Minimising the impact of bugs
  - – Minimising human error!

Ariane 501

A
cautionary
tale

Bugs: Ariane 5 flight 501

- Cost
  - – $500 million of satellites on board
- The bug
  - – “Type conversion error” (Jargon!)
  - – A 64-bit number was converted to a 16-bit number
  - – The value of the horizontal position was lost
  - – Ariane self-destructs correctly
- The error
  - – Code not meant for that flight?

A happy F-16

Well, almost …

In simulation, software inverted aircraft as it crossed the equator

Less dramatic but..

- On August 28, 1993, 2 a.m. clocks in some PCs in Israel suddenly lost an hour.
- On October 24, 1993, at 2 a.m. some PCs in the UK did *not* lose an hour. Unfortunately everyone else was turning back their clocks that morning.

Computer Bug

- Unwanted property of program code or hardware
- Especially when it causes a malfunction
- Bugs are common
  - “In Windows 98 Microsoft supposedly fixed 3000 bugs.” *PC Computing, Sept. 1998*
  - Bugs can be unwanted security holes

First Bug?

- Moth found in the Mark II computer by Admiral Grace Hopper in 1947

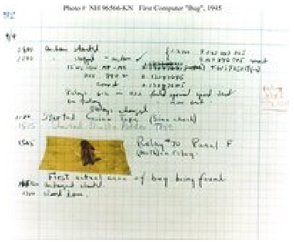


Photo # NH 96566-KN First Computer "Bug", 1945

Started Cosine Tape (Sine check)

Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

Remember..?

- **Ariane:** Programme was doing the right thing in the wrong rocket – error in <u>requirement</u>
- **Summertime:** Programme was correctly doing the wrong thing – error in <u>specification</u>
- **F-16, Mariner:** Programme(r) made a mistake – error in <u>implementation</u>

Software design process

- **Requirements:** statement of the problem
  - Validation (fails: Ariane 501)
- **Specification:** statement of what to do
  - Verification (fails: date error)
- **Implementation:** doing it
  - Design, Testing (fails:f16 (nearly))
- Note: the F-16 bug was the only one caught

Early bug: IEFBR14

- IEFBR14: one line of code for an IBM mainframe computer used in 70’s
- Instruction of code:
  - “Do nothing” (i.e. wait for a short time)
- Contained a bug!
  - Forgot to prepare the memory for the next instruction
  - Subsequent instructions went wrong
- Fixed code increased code size to 4 bytes!

A few other causes

- Evolutionary bugs (requirement drift)
  - Ariane, Patriot missile

- Human Interactions
  - USS Yorktown (data entry error), HMS Sheffield (operational errors)

- Communication
  - Mars Orbiter: mixed imperial and Metric units
- Most major failures have multiple causes

Bugs: Patriot missile

- Error calculating the time since the computer booted
  - Binary representation of 0.1 seconds limited to 24 bits
- Once activated, navigation system drifts
- In the Gulf War 1991
  - Caused a patriot missile to fail to intercept a Scud missile
  - 28 killed, 100 injured

Computer programs

- Computers are excellent at following instructions
  - Identify how to solve the problem
  - Use a computer!
- Major difficulties are
  - Expressing problems that can be solved using efficient *algorithms*
  - Giving the computer the correct instructions
  - Making the program user-friendly

Bugs in programs

- Memory leak
  - Forget to release memory after it has been used
- Other easy/common mistakes
  - Variable not set to the right initial value
  - Divide by zero: answer is infinity!
  - Get a number wrong by 1
  - Loops that never end
- Spelling mistakes
  - Usually prevented by the code not compiling
  - Not always! (Mariner 1)

Mariner 1:

- Failed “because the line
  - **`DO 10 I=1.100`** should have read
  - **`DO 10 I=1,100”`**
- There’s rather more to it than that..

Fault tolerant systems

- Creating fault *free* systems
  - Difficult and time-consuming
- Fault *tolerant* systems operate successfully despite faults
- Hardware: back-up systems
- Software:
  - Keep multiple copies of (back-up) the data
  - Identify and monitor critical variables
  - Checkpointing: reset system to a stored set of values

Example: Aircraft failure rates

- Fatal accident rate

  1 death in 1,000,000 flying hours
- System causes 10% of accidents
- 100 critical systems in an aircraft
- Rate of failure

  1,000,000 hours × 100 systems / 10%

  = 1 fatal fault in 1,000,000,000 system flying hours

  Good enough?

Software design: Waterfall model

Analyse the problem
→ Design solution architecture
→ Design solution details
→ Write program code
→ Test code
→ Maintain code

- Problems:
  - Original analysis is difficult
  - Problems identified at the end can be expensive to fix

Iterative design model

- At each stage
  - Design → Prototype → Evaluate → Redesign
  - All stages developed concurrently, with feedback between all stages
- Advantages
  - User-defined from the start
  - Performance can be measured much earlier
- Problems
  - Time consuming
  - Requires good management

Defensive programming

- Anticipate possible circumstances
- Trust nothing
  - Check what you are being told e.g.
    - angles between 0 and 359º
    - day-of month is between 1 and 31
  - Check what you are telling others
  - Sanity checks on actions taken
- Fail in predictable manner if fault occurs
- Layered protection including hardware 'back-stops'

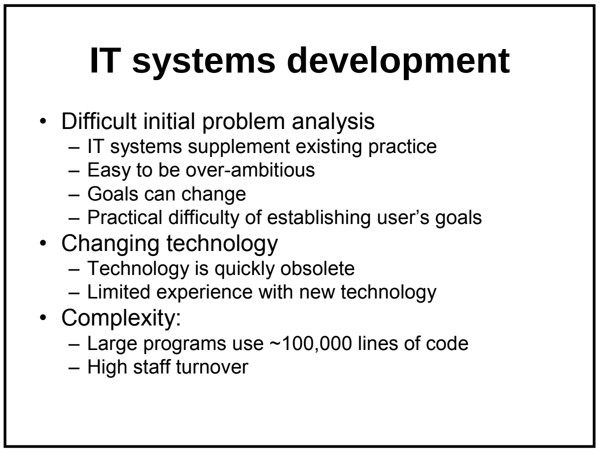
Beta testing

- Refers to the 2nd phase of software testing
  - Sample of the intended audience tests the product
  - It works for the programmer, does it work for the user?
  - Provides a "preview" of software: it's free! Buggy!
- Emerging software: look for "Beta" versions
  - At Google http://labs.google.com/
  - At MSN now: http://beta.search.msn.com/
  - Dedicated Web site www.betanews.com/
- [Beta is the second letter in the Greek alphabet. "Alpha" testing refers to the first phase: checking it works for the programmer]

In the news …

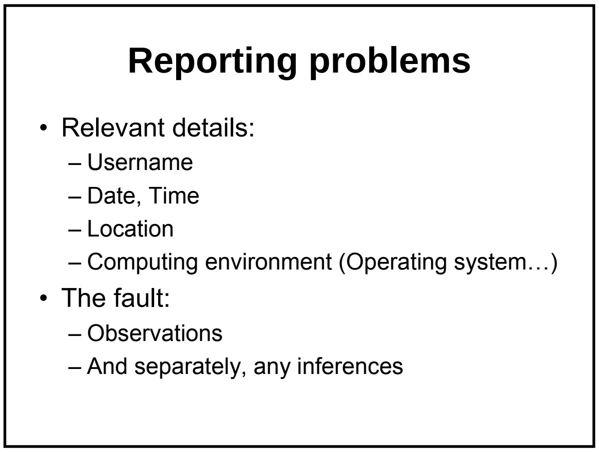
- Cost of Child Support Agency's new computer system:
  - £456 million

  (Scottish parliament building: £431 million)
- Unable to cope with the work load
  - Backlog of 30,000 cases per month
- How could this happen?
- Source: http://news.bbc.co.uk/1/hi/uk_politics/4020399.stm



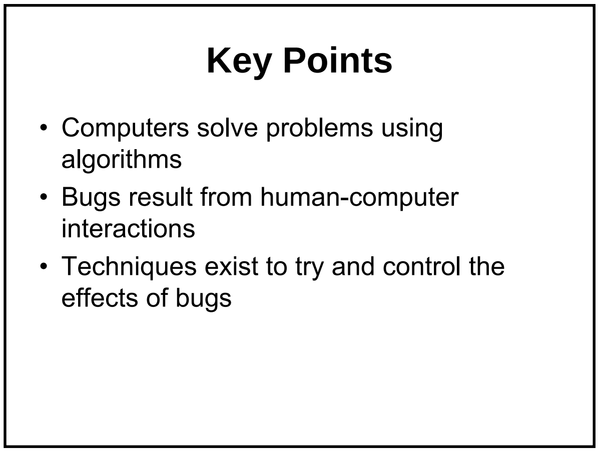
IT systems development

- Difficult initial problem analysis
  - IT systems supplement existing practice
  - Easy to be over-ambitious
  - Goals can change
  - Practical difficulty of establishing user’s goals
- Changing technology
  - Technology is quickly obsolete
  - Limited experience with new technology
- Complexity:
  - Large programs use ~100,000 lines of code
  - High staff turnover



Reporting problems

- Relevant details:
  - Username
  - Date, Time
  - Location
  - Computing environment (Operating system…)
- The fault:
  - Observations
  - And separately, any inferences



Key Points

- Computers solve problems using algorithms
- Bugs result from human-computer interactions
- Techniques exist to try and control the effects of bugs