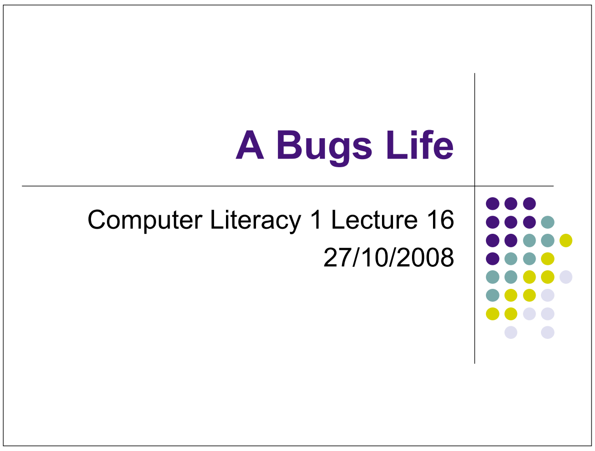
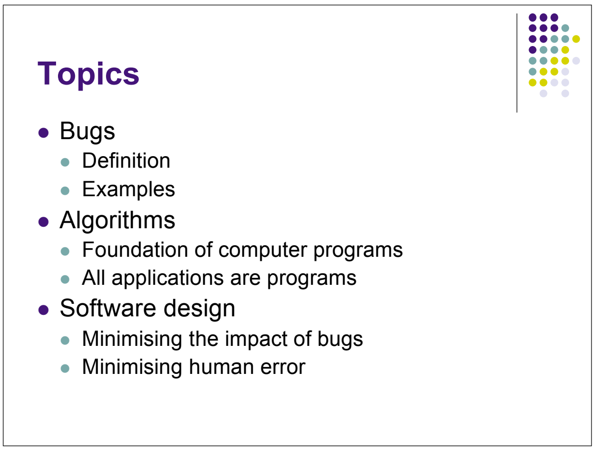


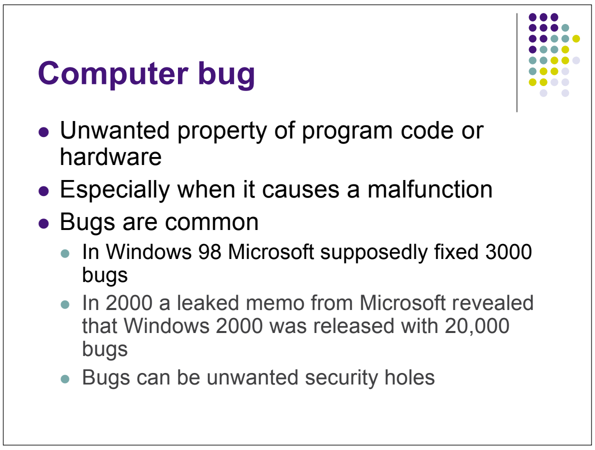
A Bugs Life

Computer Literacy 1 Lecture 16
27/10/2008



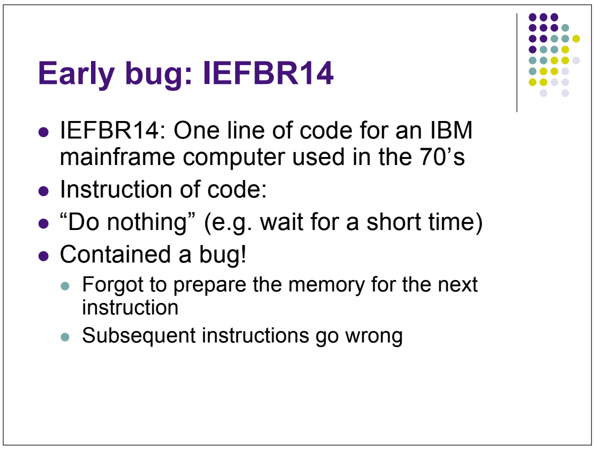
Topics

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



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
  - In 2000 a leaked memo from Microsoft revealed that Windows 2000 was released with 20,000 bugs
  - Bugs can be unwanted security holes



Early bug: IEFBR14

- IEFBR14: One line of code for an IBM mainframe computer used in the 70’s
- Instruction of code:
- “Do nothing” (e.g. wait for a short time)
- Contained a bug!
  - Forgot to prepare the memory for the next instruction
  - Subsequent instructions go wrong

Bugs: Patriot missile

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

➔ 28 people were killed and 100 injured

Computer programs

- Computers are excellent at following instructions
  - Follow your command literally
  - Can solve problems quickly
- Major difficulties are
  - Expressing problems that can be solved by efficient algorithms
  - Giving the computer the correct instructions
  - Making the program user friendly

Bugs in programs

- Memory leak
  - Forget to release memory after it had been used (e.g. IEFBR14)
- Other easy/common mistakes
  - Variable not set to the right initial value
  - Loops that never ends
- Spelling mistakes
  - Usually prevented by the code not compiling
  - Not always (Mariner 1)

Bugs: Ariane 5 flight 501

- Cost
  - $500 million of satellites on board
- The bug
  - "Type conversion error"
  - A 64-bit number was converted in a 16-bit number
  - The value of horizontal position was lost
  - Ariane self-destructs correctly
- The error
  - Code not meant for that flight?

Ariane 5 Flight 501

- http://www.youtube.com/watch?v=IONcgYzVFIg
- Year was 1996

Software bug halts F-22 Flight

- On February 11, 2007 twelve raptors flying from Hawaii to Japan were forced to turn back because of a software glitch
- Their computers crashed when they crossed the international date line!
- They had to turn around an follow their tankers by visual contact back to Hawaii

Less dramatic but happened

- On August 28, 1993, 2a.m. clocks in some PCs in Israel are suddenly loosing an hour
- On October 24, 1993, at 2a.m. some PCs in the UK don't turn back their clocks like they were supposed to

Mariner 1

- Mariner 1 should have been an spacecraft on a Venus flyby mission
- Instead a security officer called its destructive abort 293 seconds after its launch
- It's claimed that the bug was a single sign in the code that was wrong:

`DO 17 I = 1.100` should have been

`DO 17 I = 1,100`

Remember

- **Ariane**: Program was doing the right thing in the wrong rocket - error in requirement
- **Change from summer to winter-time**: Program was correctly doing the wrong thing - error in specification
- **F-22, Mariner**: Programme(r) made a mistake - error in implementation

Software design process

- **Requirements**: statement of the problem
  - Validation
- **Specification**: statement of what to do
  - Verification
- **Implementation**: doing it
  - Design, Testing

When it all goes wrong

- **Fault** - an error lurking in the program
- **Error** - fault is triggered
- **Failure** - program takes inapproriate action as a result

Fault tolerant systems

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

Software design: Waterfall model

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

Iterative design model

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

Beta testing

- Refers to 2nd phase of software testing
  - Sample of intended audience test the product
  - It works for the programmer, does it work for the user?
  - Provides a “preview” of software
- Dedicated website: www.betanews.com

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 of code
  - High staff turnover

During the implementation

- Monitoring calls with business
- Schedule of events checking
- Formal checkpoints
- Business checkout
- Incident management. Formal control of any problems
- Go / No Go decision
- Ensure all in place for staff to use

Post implementation

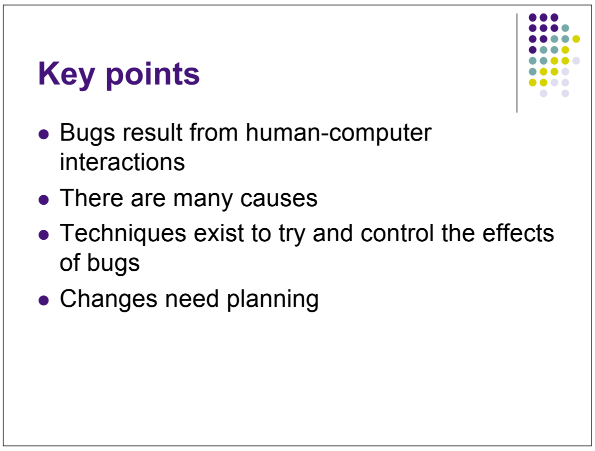
- Analysis of any problem
  - What was their problem?
  - What was done to resolve them?
  - Are any further fixes needed?
- Monitoring of ongoing system performance
  - Are the transactions being processed correctly?
- How is the business getting on with the system?
  - Has it been well received?
  - Is everyone able to use it easily?
  - Any further action needed?

London Ambulance Fiasco 1992

- The London Ambulance (LAS) Computer Aided Dispatch failed dramatically on October 26 1992 shortly after it was introduced
  - The system could not cope with the load placed on it by normal use
  - The response to emergency calls was several hours
  - Ambulance communications failed and ambulances were lost from the system

LAS Fiasco

- A series of errors were made in the procurement, design, implementation, and introduction of the system.
  - There appears to have been NO backup procedure at all
  - Design of user interface was inadequate
  - No consideration was given to system overload



Key points

- Bugs result from human-computer interactions
- There are many causes
- Techniques exist to try and control the effects of bugs
- Changes need planning