2006-

“Designing and implementing Efficient Tests and Test Strategies”

by

Les Hatton

Professor of Forensic Software Engineering,
CISM, University of Kingston, UK
lesh@leshatton.org

Version 1.1: 14/Feb/2006

How good is good ?

A useful criterion

- Define a defect as a fault that has failed
- Define an executable line of code as any line of code which generates an executable statement
- Define asymptotic defect density as the upper bound of the total number of defects ever found in the product's entire life-cycle divided by the lines of code

If your asymptotic defect density is < 1 defect per KXLOC (thousand executable lines of code), you are doing about as well as has ever been achieved.

How bad is bad ?

NIST (US National Institute of Standards and Technology)

- 2002 report estimating costs of software failure in US alone at $60 billion per year
- 80% of software development costs are finding and fixing defects
  - Economist Science Technology Quarterly 19/Jun/2003

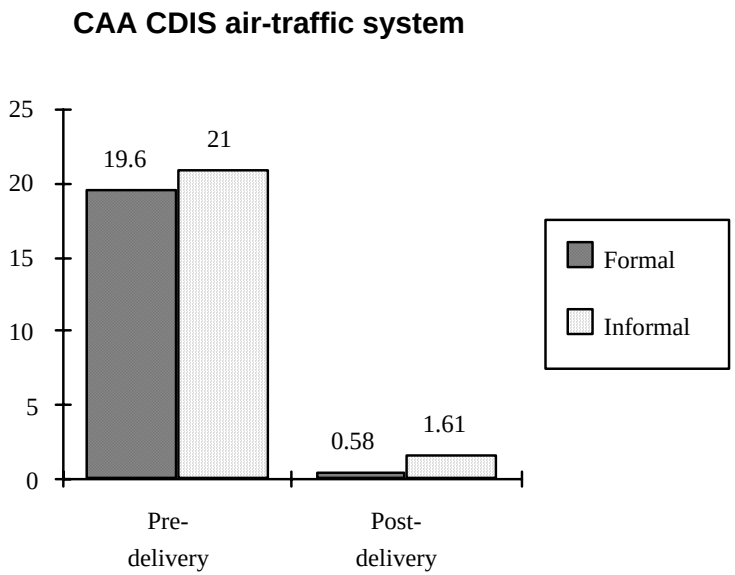
Overview

- **Where do we start ?**
  - Aristotleans v. Babylonians: the role of measurement
  - Some clues from air-traffic control
  - Typical defect profiles
- **Complicating factors**
- **Techniques**
- **Some interesting questions answered**

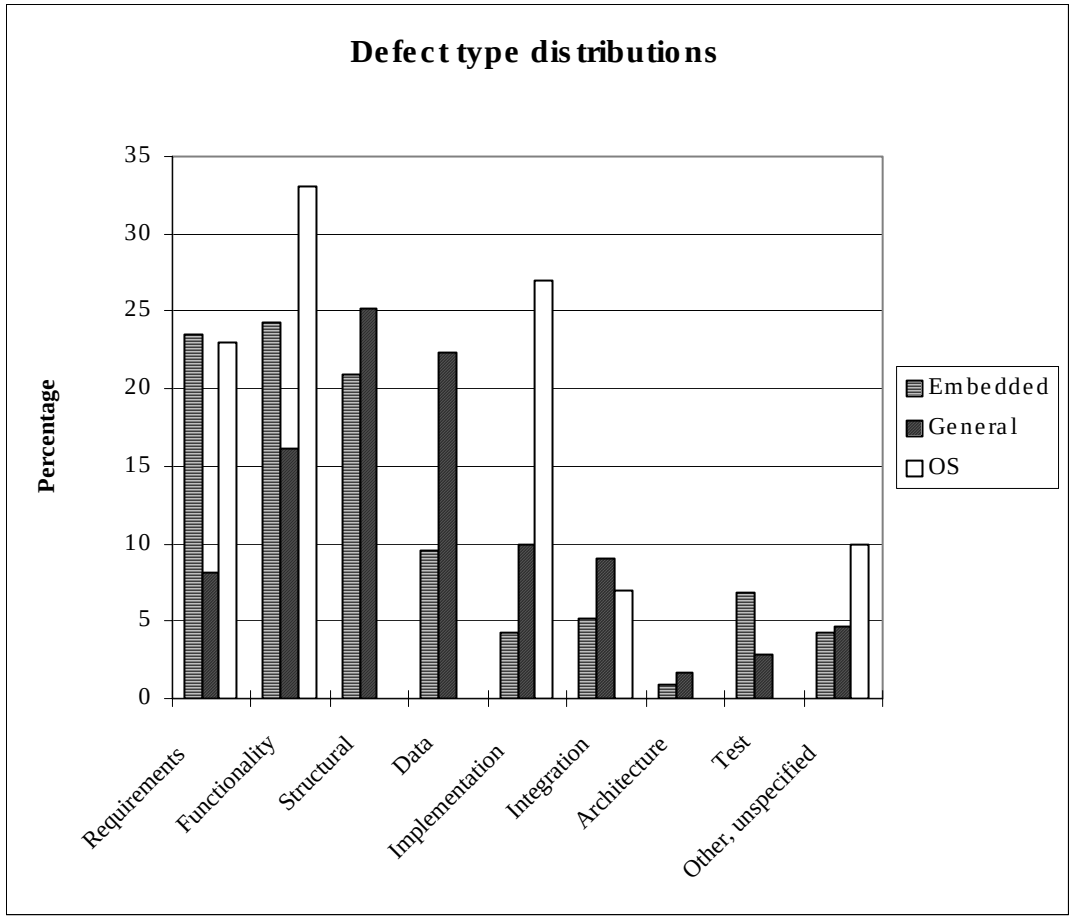
Aristotleans v. Babylonians

- **Aristotleans …**
  - Decide everything by deep thought
- **Babylonians …**
  - Learn the hard way by sticking their fingers in light sockets.

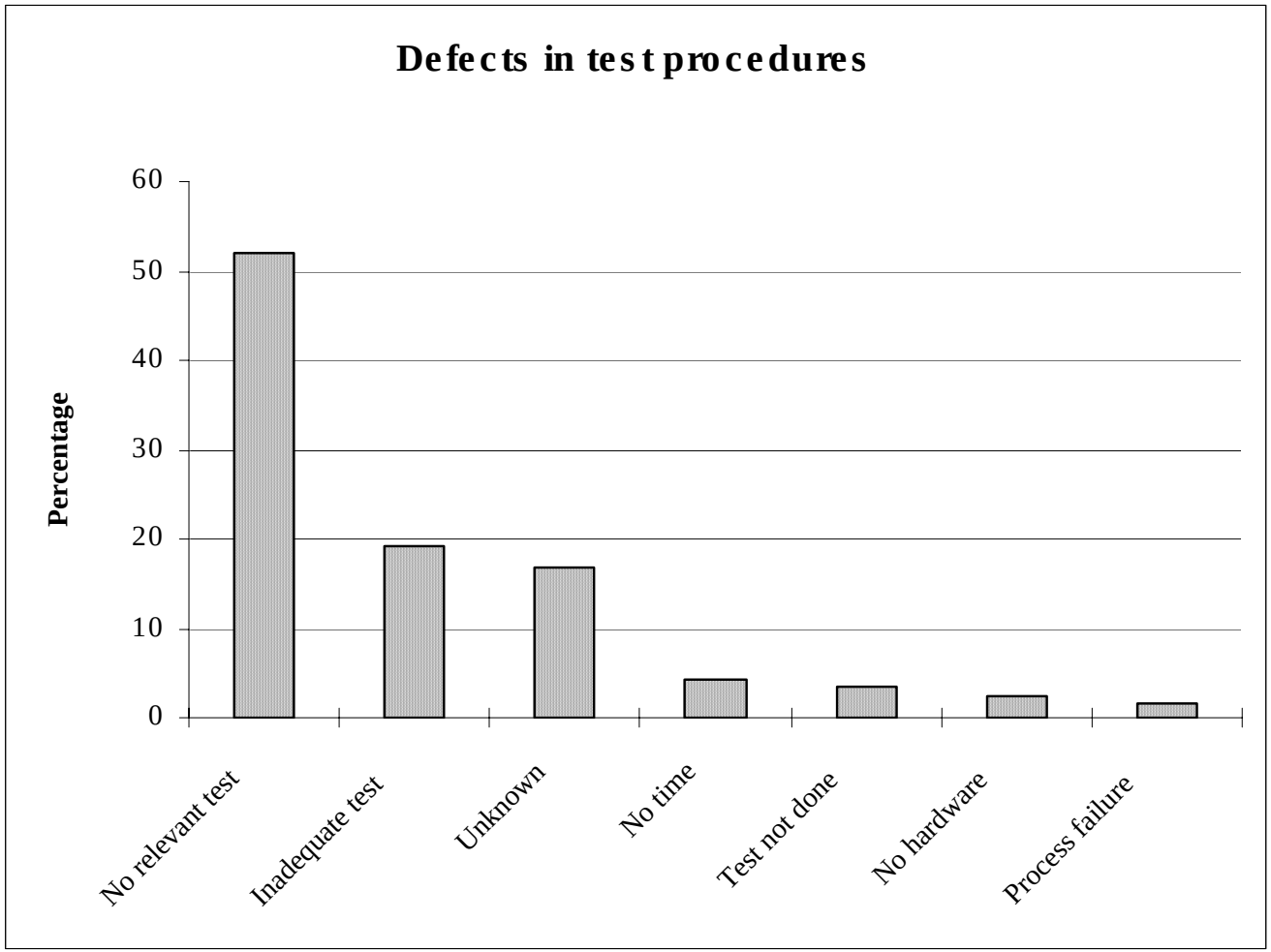
Some clues from air-traffic control



Typical defect profiles



Defects in test procedures - OS



Overview

- **Where do we start ?**
- **Complicating factors**
  - Architecture
  - Complexity
  - Long delay defects
  - Education
  - Spreadsheets
  - Imprecise specifications
- **Techniques**
- **Some interesting questions answered**

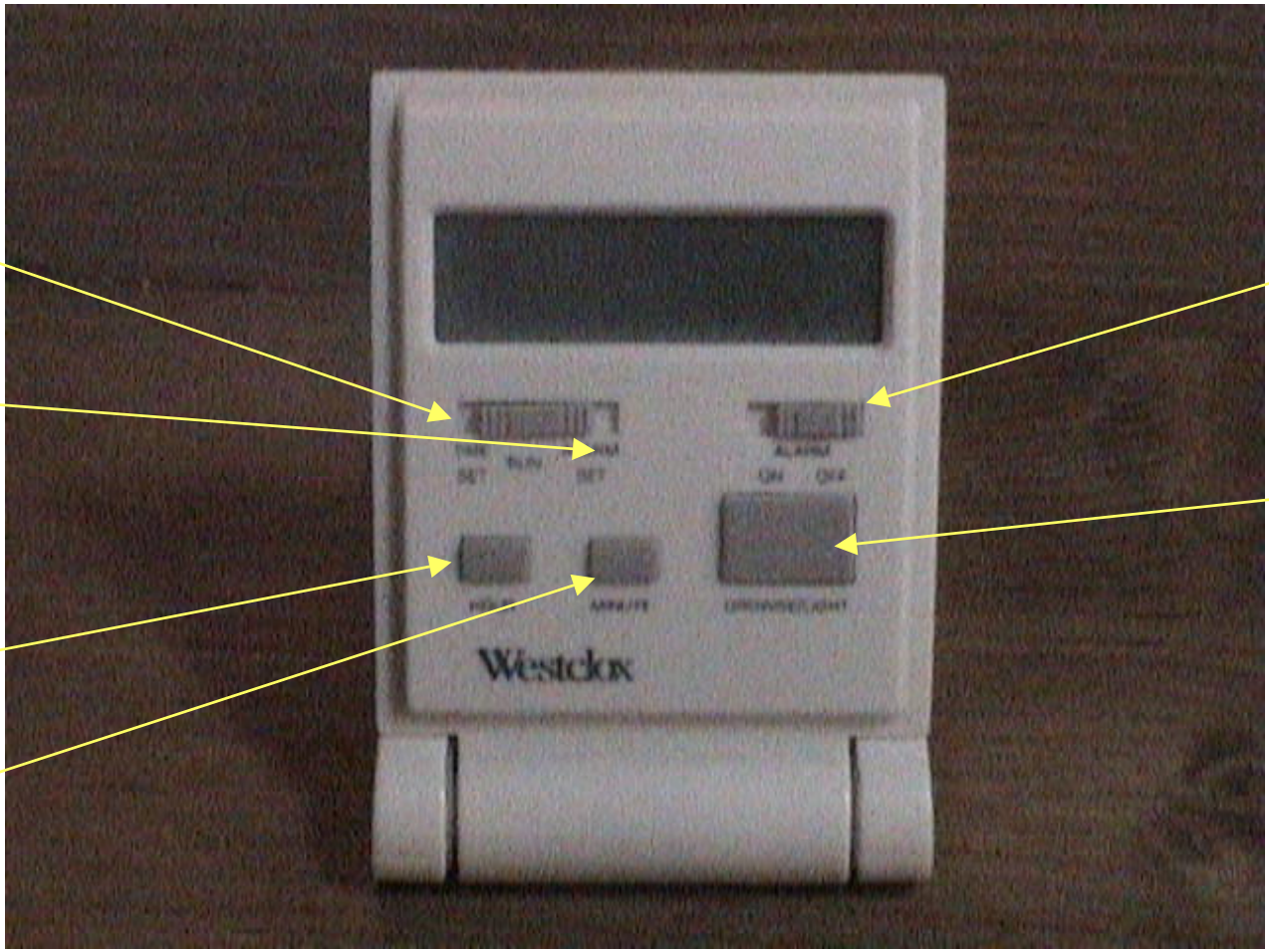
Architecture

- **Some architectures are much more testable than others**
  - Good
    - Client / Server
  - Not so good
    - GUI
    - Embedded systems
  - Terrible
    - Real-time, high-integrity systems

A tale of two alarm clocks

- **Alarm clock 1**
  - Purchased 5 years ago and faultless ever since.
  - Staggeringly simple and intuitively obvious interface which has never required the instructions to be consulted.

A tale of two alarm clocks – the sublime



Time set

Alarm set

Hour

Minute

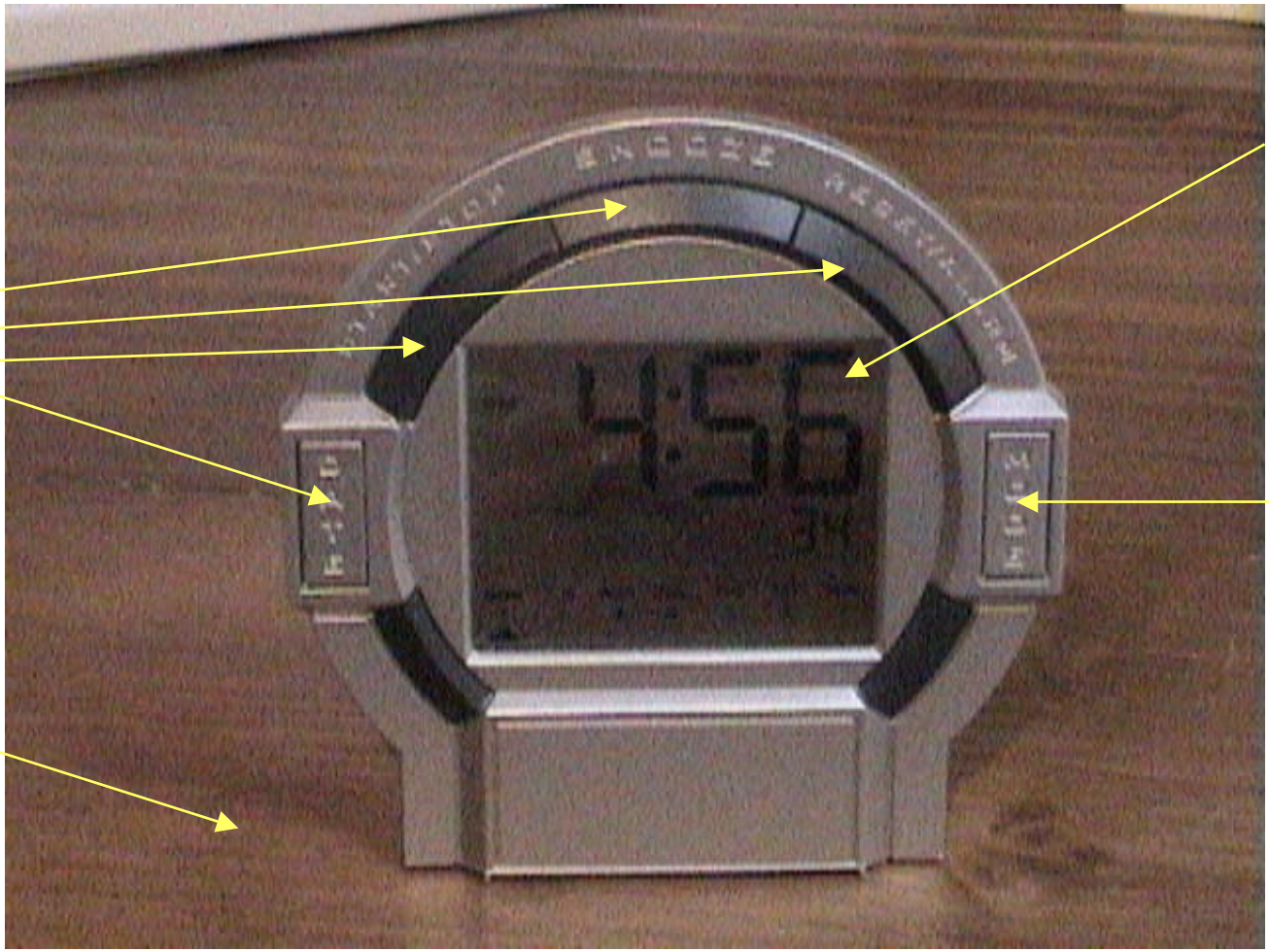
Alarm On/off

Drowse

A tale of two alarm clocks

- **Alarm clock 2**
  - Purchased 1 year ago and frequently resets itself
  - Staggeringly complex and intuitively non-obvious interface.

A tale of two alarm clocks – the ridiculous



More buttons

Random numbers

Mode - ARGH !

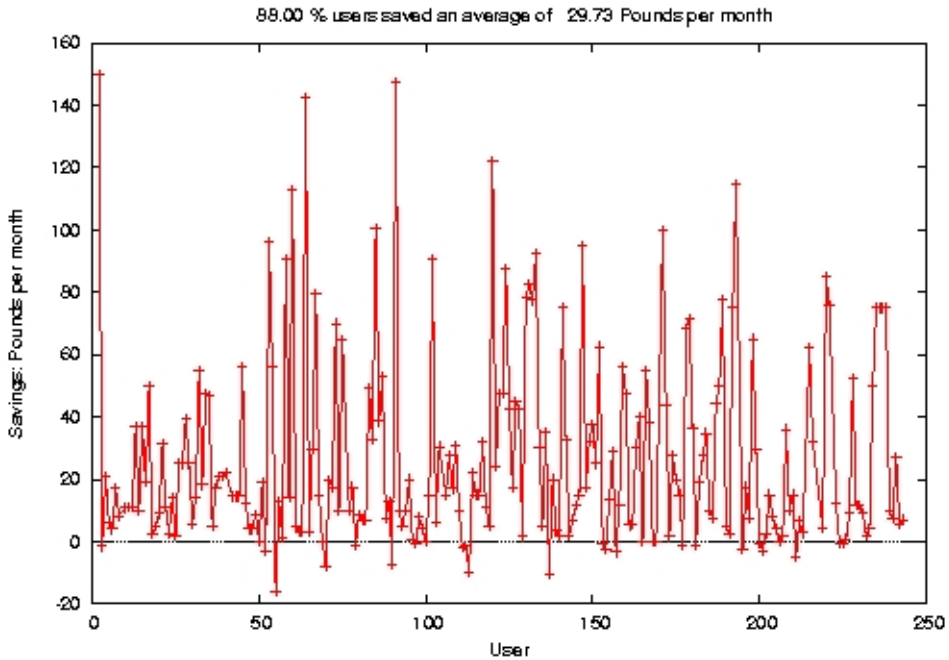
Desk

A tale of two alarm clocks

- **Alarm clock 2 – some examples**
  - Time setting
    - Press MODE button THREE times. Then press SELECT and SET repeatedly.
  - Alarm setting
    - Press MODE button TWO times. Then follow time setting.
  - Hourly chime on/off
    - Press SELECT and MODE button *simultaneously*
  - Alarm on
    - Press DATE and SELECT button *simultaneously*
  - Stop watch / lap counter
    - Press MODE button ONCE and then beg for mercy as clock gibbers with entertaining series of random beeps.
  - Turning alarm off
    - Hurl out of window after standing on clock screaming.

Obfuscation in mobile phone charges UK Results 2Q04

Illustrates the natural growth of complexity *when technology allows*



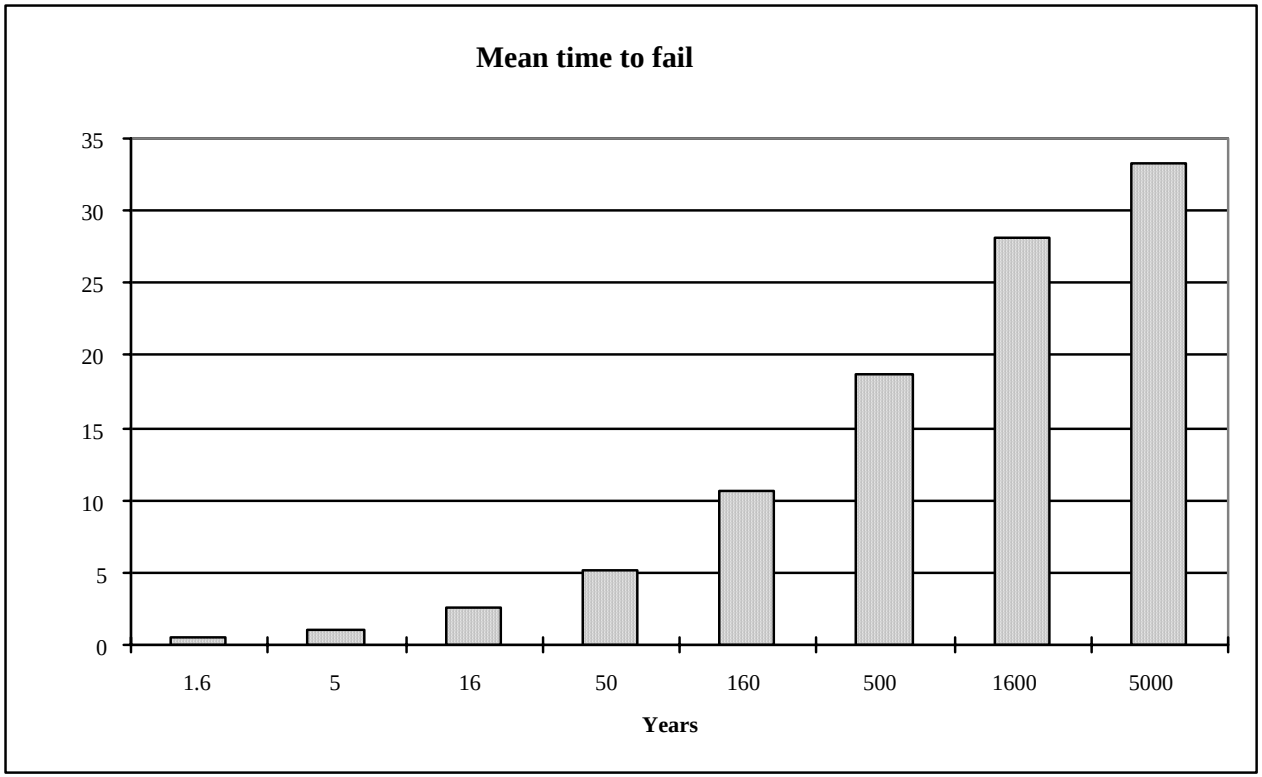
Average saving £29.73 per month

Fuzzy global optimisation web server finding minimum phone charges

http://www.betterdeal.co.uk/

Long-delay defects

% faults that failed



Long delay defects

Avionics …

- 28/Jul/2003. “As recently as February, test pilots of the new F/A-22 (Raptor) fighter were spending an average of 14 minutes per flight rebooting critical systems. This is now down to only 36 seconds per flight.

  Washington Post.

Education: An interesting survey

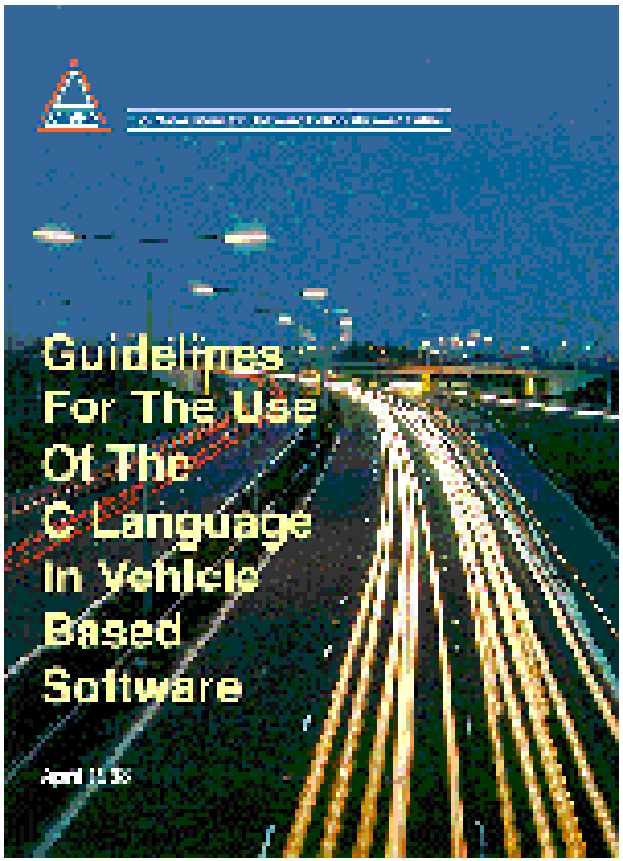
- **Debugging is a close neighbour of testing. Andrews (2002) found the following in a recent survey of debugger use:-**
  - By far the most common debugging tool in both academic and commercial users is the print statement
  - About 20% of all respondents had never heard of static checking
  - About 50% of all respondents had never heard of slicing to show dependencies between statements
  - Of 12 well-known debugging techniques, half of the respondents had never heard of half of them.

Andrews, M. (2002) Ph.D thesis, University of Kent, UK

Spreadsheets

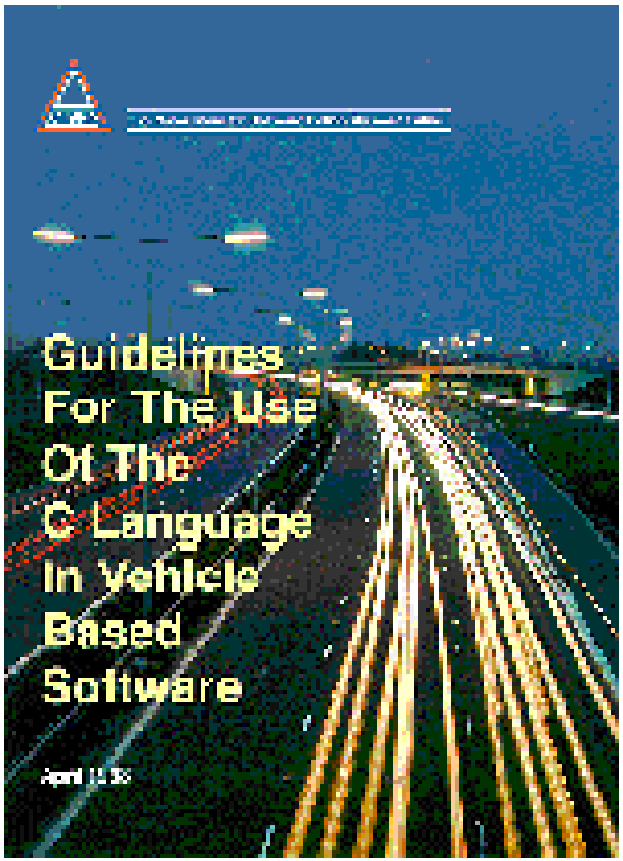
- **One of the great liberating applications of the 80s and 90s**
- **One of the major headaches of the 21st century**
  - Weird arithmetic
    - -x^2+1 != 1-x^2 (Allan Stevens 2005)
  - People keep data in them instead of in databases
    - This a major fly in the ointment in most companies because people cannot exchange data.
  - They are hard to test and consequently full of defects
    - 90% of all spreadsheets had errors which led to more than 5% error in the results. Ray Panko (University of Hawaii)

The MISRA C standard



- **In April 1998, the Motor Industry Software Research Association (MISRA) published a set of C guidelines for use in vehicle-based software.**
  - 93 rules + 34 guidelines
  - Consistent with development to SIL3
  - Sensible deviation policy
  - Some uncertain areas
  - The standard is cross-referenced against the ISO C 9899:1990 standard for traceability

The MISRA C standard



- **The test suite was constructed after the standard was published**

  24% of the rules were sufficiently imprecise that test cases could not be constructed at all or could not be constructed without breaking other rules.

  (http://www.leshatton.org/ for more details)

  Tests should be designed concurrently with specifications to make sure they are feasible.

Overview

- **Where do we start ?**
- **Complicating factors**
- **Techniques**
  - Tool-building
  - Balancing static and dynamic techniques
  - Standard attacks
- **Some interesting questions answered**

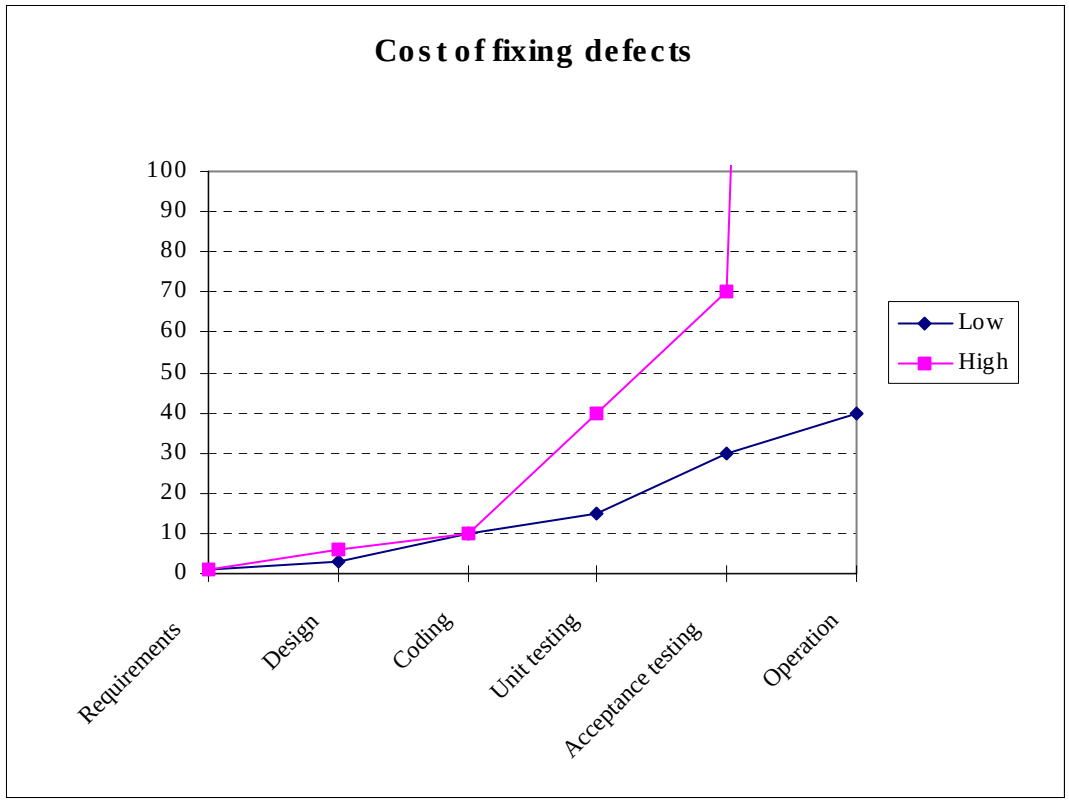
Open source tooling

- **Scripting languages**
  - Portable (reducing test setup costs substantially)
  - Expressive (reducing test implementation costs)
  - Free (you can probably work this bit out)
  - Examples
    - Tcl/Tk
    - Perl
    - Python

Open source tooling

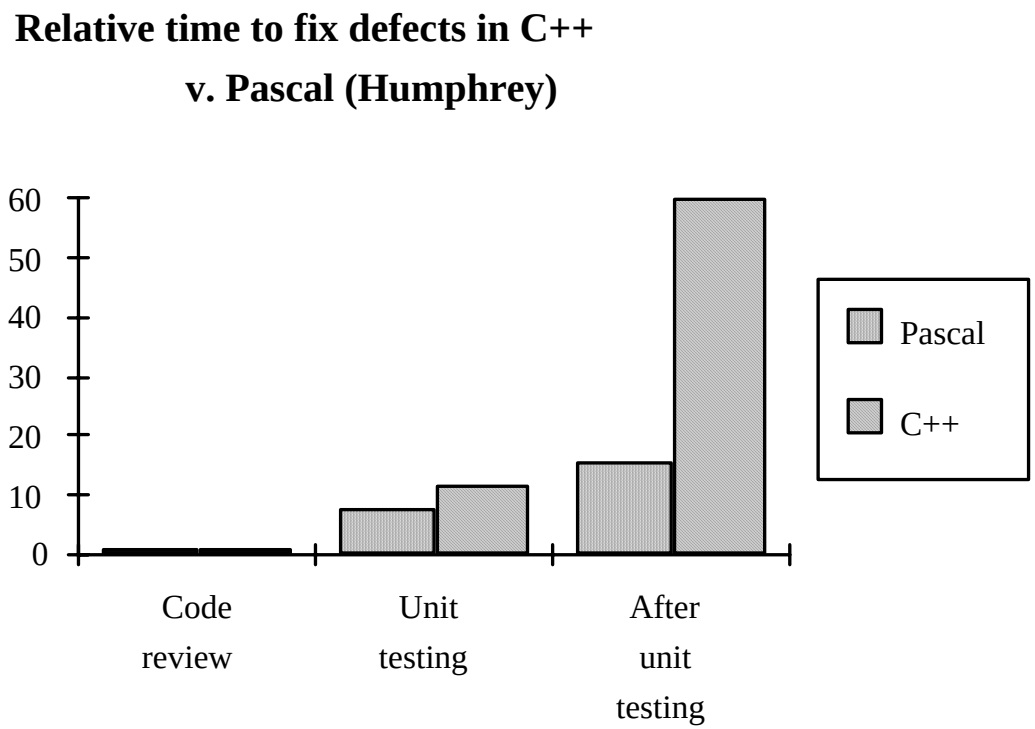
- **Scripting languages**
  - An example follows of a portable test harness written in Tcl/Tk, (about 700 lines of harness including all graphics). This script was developed on Linux and simply moved to Windows along with the regression oracle, (which are simple text files).
  - <u>Start Gundalf regression</u>

Cost balance between static and dynamic testing



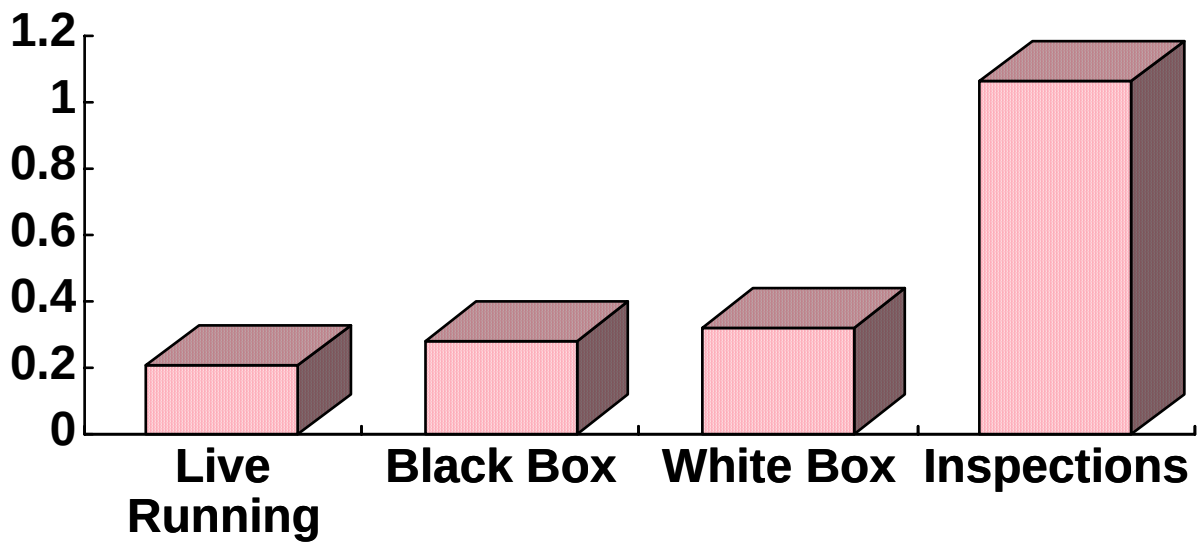
Embedded systems tend to follow the high curve.
Data from Boehm, (1981).

Testing in OO systems



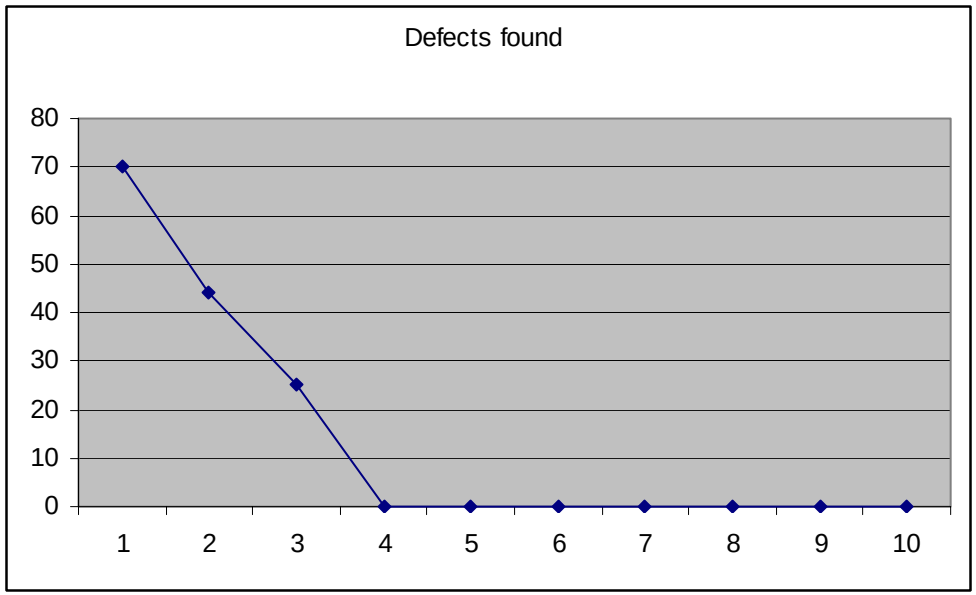
What do we know about inspections ?

Consider the following rates of defects found per hour, (Grady & Caswell, (1987)):



Inspections - how do you do them ?

Defect density found



Inspection speed (100s LOC/ hour)

These measurements are due to Humphrey (1995) made on 25 C++ projects

Standard attacks

- **Whittaker's standard attacks (2003)***
  - User interface attacks (17)
    - Exploring inputs (6)
    - Exploring outputs (4)
    - Exploring stored data (3)
    - Exploring computation and feature interaction (4)
  - System interface attacks (6)
    - Media-based attacks (3)
    - File-based attacks (3)

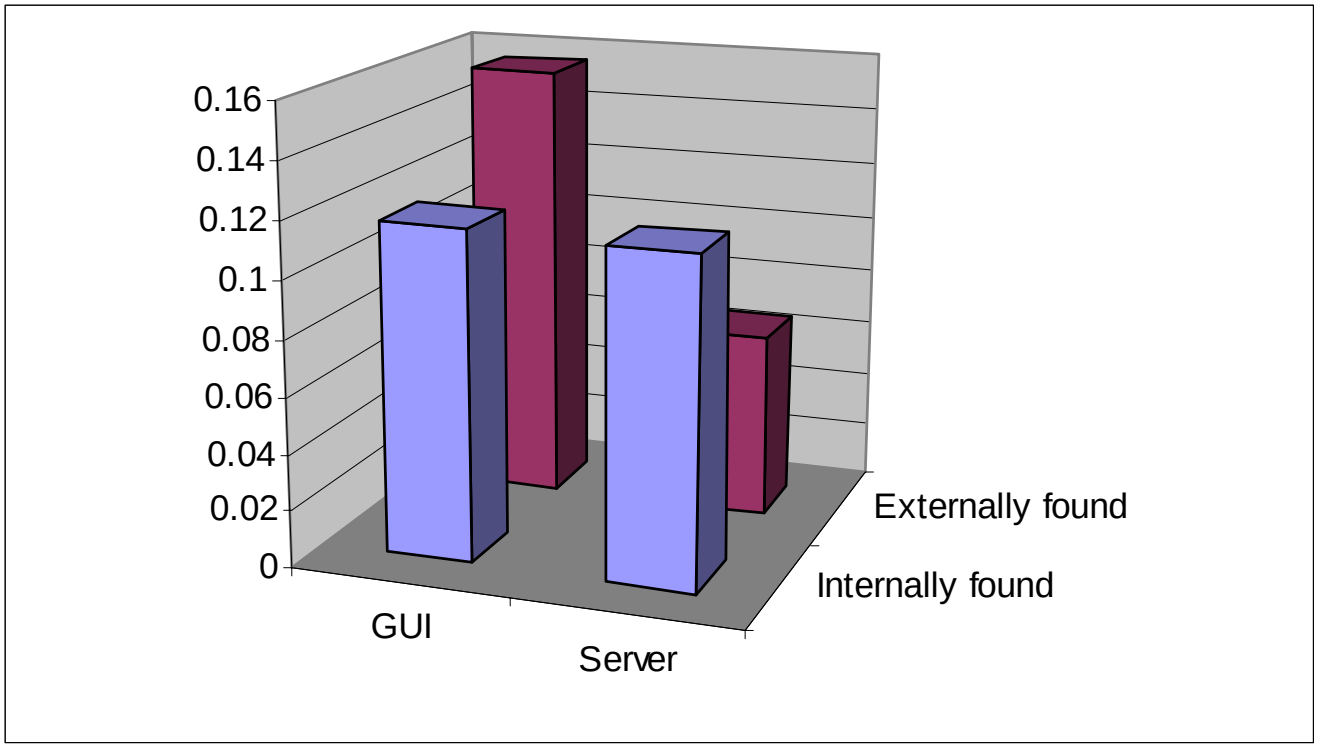
James A. Whittaker (2003) "How to break software", Addison-Wesley, ISBN 0-201-79619-8

Overview

- **Where do we start ?**
- **Complicating factors**
- **Techniques**
- **Some interesting questions answered recently**
  - Should we stop testing after delivery ?
  - Do checklists help inspections ?

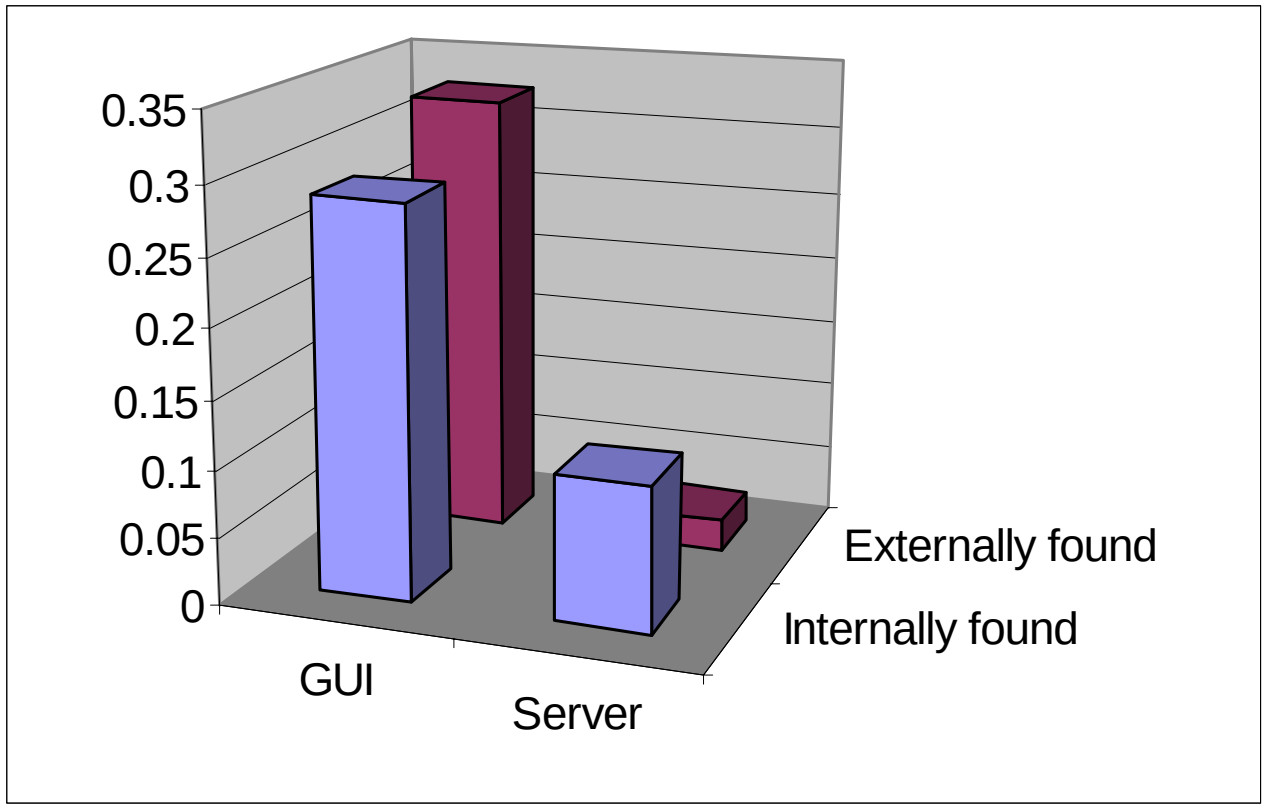
Testing after delivery: case history 1

Asymptotic defect density (defects per KLOC)



Testing after delivery: case history 2

Asymptotic defect density (defects per KLOC)



Parallel inspections

- **These can be used to test use of checklists for example:**
  - 22 2xperson teams in India, Germany and Austria half using checklists and half without
  - Single person inspections followed by 2xperson team comparison to find ***common*** defects
  - If P(A) is probability that A finds defect and P(B) is probability that B finds defect, then condition is for independence is P(A^B)=P(A)P(B).
  - This could not be rejected at 5% level.
  - Checklists are therefore having no significant effect.

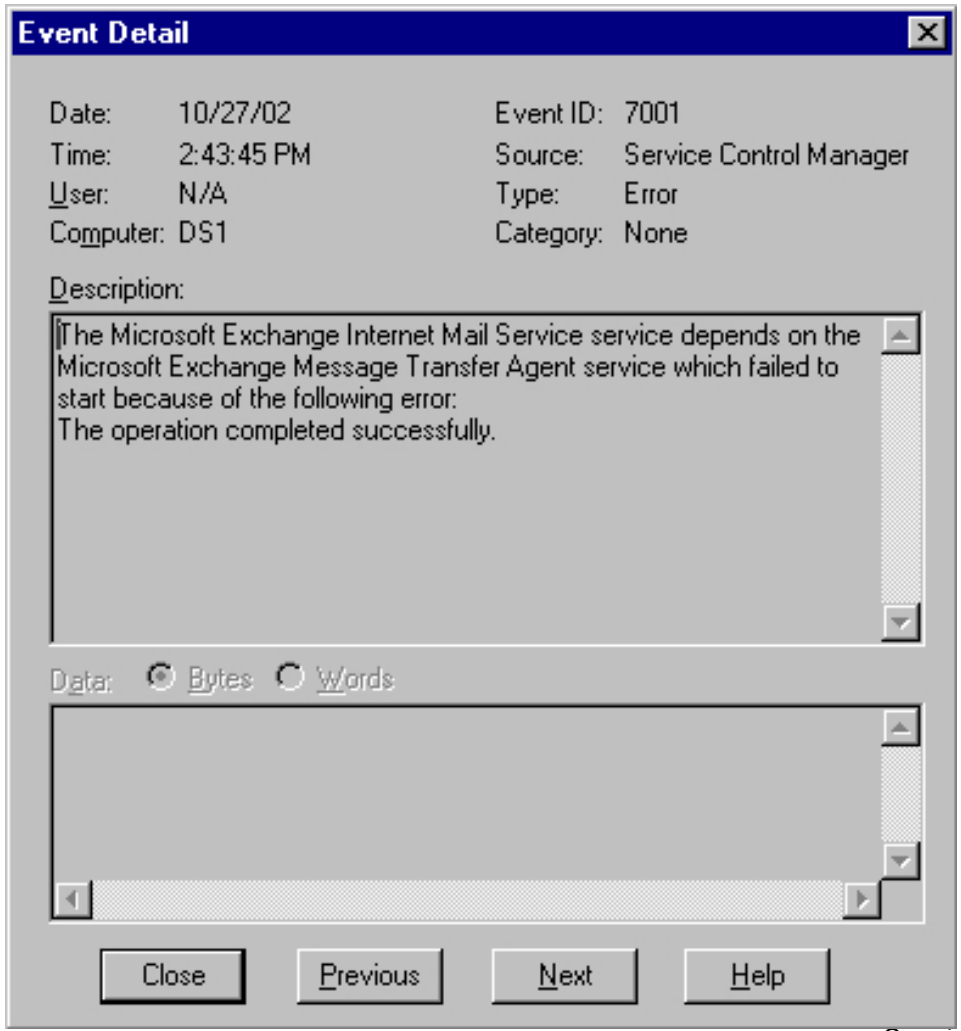
Experimental results

- **Improving test strategy efficiency requires:-**
  - Awareness of the properties of failure and the intended use and ubiquity of the product
  - Awareness of how good is good
  - Better education
  - Very early involvement with design and specification
  - Continuing to test after a product is released, (assuming product updates are economic to do)
  - Test and defect measurements and the patterns concealed in them.

General conclusions

- **Improving test strategy efficiency requires:-**
  - Awareness of the properties of failure and the intended use and ubiquity of the product
  - Awareness of how good is good
  - Better education
  - Very early involvement with design and specification
  - Continuing to test after a product is released, (assuming product updates are economic to do)
  - Test and defect measurements and the patterns concealed in them.

A dehumanising message



For further information …

This paper and many other downloadable papers and pieces of software can be found at:-

http://www.leshatton.org/